SPECIAL • LANS vs MULTI-USER SYSTEMS

COMPUTER SHOPPER®

APRIL 1987
VOL. 7 NO. 4
ISSUE 87
$2.95
CANADA $3.50
02927

Money Saving Software • In-Depth Articles

Introducing AppleShare™ For AppleTalk™ Network

Spotlight On Legal Software And Services

New GFA BASIC For Atari ST

America's Mailorder Headquarters
Lyco Computer
World's Leader in Sales and Service
See Pages 10 & 11

80386 COMPUTER
$3195.00
NDRC
214-368-4994

COMMUNICATIONS ELECTRONICS INC.™
25¢ Diskettes
800-USA-DISK
See ad index

Premium Diskettes!
MEI/Micro Center — See Our Full Page Ad for Details
1-800-642-1148
25¢
DS/DD with Sleeves

100 FREE PROGRAMS!
FOR THE IBM-PC/XT • WHEN YOU BUY A BOX OF OUR DISKS!
110 DS/DD 5¼" DISKS WITH SLEEVES & LABELS $49.50/CASE
UNITECH (800)343-0472 24 HOURS A DAY 7 DAYS A WEEK!

BEST PRICES - BEST PRODUCTS - BEST SERVICE
MW MidWest Micro-Peripherals
1-800-423-8215
29 FULL PAGES OF SAVINGS
SEE AD INDEX

American Semiconductor
1-800-237-5758
TURBO XT $399!
10 MB KIT $199!
256K...$1.99!

BCE 70% Off Most Items!
See The Red Section!

PRINTERS Plus
800-562-2727

EGA EXCELLENCE, CALL!
Proto PC, Inc. 612-644-4660

See Advertiser's Index Page 4

EB0260T

COMPUTER SHOPPER®

TABLE OF CONTENTS

INTRODUCING APPLESHARE™ FOR APPLETALK™ NETWORK

- **Spotlight On Legal Software And Services**
- **New GFA BASIC For Atari ST**

ARTICLES
LANS vs Multi-User 5
A LAN Primer 5
Editorial 8
Proteon's Networking Products Give Spectacular Performance 12
The Saga of Manny The Power Moocher 18

IBM-MS DOS
Equity III+ Has Increased Performance 99
Wash and HotDOS! Hotdog! More Freeware Utilities 99
Hard Disk Management Simplified 100
Recover Corrupt Data Files 169
Setting Up Foreign Languages On A PC 172
SNOBOL4: The Standard For Strings 175
SNAP: Not Just Another Pretty Interface 180
Line Numbering Of WordStar Document Files 182
It's Leading Edge WP For Me! 186
Confessions of a Low Level Formatter 188
Customizing Your Turbo Pascal System 191
Stitch Grapher Creates Cross-Stitch Designs!!! 194
Streamline The Paper Jungle With Formtool 194
Reference Set 196
Compaq Lowers Compaq DeskPro Price 196
X-Lock—Controls Access To PC 200
Turbo Pascal Quiz Programs 204
Son of Market Trend Anaylzer 205
Word Certain 1.00—The Ultimate Word Processor? 209

MACINTOSH
Apple Provides Network With AppleShare File Server 103
Industry Supports Apple Communications Products 103
SuperMac Software Announces SuperLaserSpool 210
Engineering Analysis Package For The Macintosh 211
Mac●Man Brings Joystick Connection To The Mac 211
InBox for Mac No Longer Requires Dedicated Server 212
Two New SCSI Internal Hard Drives From Rodime 212
MacUniverse 217
Tempo: Macintosh Macros 221
San Francisco MacWorld 224
New Version of Borland's SideKick For Mac 228
MacInTax 1986: Income Tax Program For The Macintosh 228
Maconvert—Mac Graphics To Amiga 229
DiskTop Utilities For Mac 230
New Omnis Version From Blyth Software 230
Apple/Macintosh-Compatible Mass Storage Products 236

APPLE
Revamped Apple IIe Debuts 107
Ask The Guru 107
The Apple Memory Map 236
Apple-CP/M 243
Trailblazer—Review 248
Renting Software—Boon or Bane? 250
Analog Clock 252
Copy II Plus 254

ATARI
Applying The Atari 115

ATARI ST
GFA BASIC for The ST 119
Hacking The ST 119
Math Pack #2 260
MacroManager 263
Wanderer 273

COMMODORE
Inside Commodore 123
Rumors End, More Letters, More Info 123
Getting The Most From Word Writer 128 124

AMIGA
Amiga—The Structure Of Success 127
The Amiga Workbench 127
Amiga vs Atari: Round 3 Will the "True" Atari Please Stand Up? 128
Pagesetter Desktop Publishing For The Amiga 128
DOS-2-DOS Easy File Transfer: MSDOS To AmigaDOS 287
Three From Maxisoft For The Amiga 290
Amiga Genlock Peripheral Interface to VCR 296

TANDY
Trackstar For The Tandy 1000 131
Family Roots: For Genealogy Lovers 131

SANYO
Fixing The 768K Miller Memory Modification 135
The Silver Box 135
AUTODEX—Public Domain Disk Utility 300

TIMEX/SINCLAIR
Sinclair Survival Column 139

COMPUTER BOOKS
Books On Excel Bring Help To Mac Users 143
The Information Decade—A Challenge To Anyone In Publishing 143
Top Ten Computer Book List April 1987 144
Tentative Spring List 144
Add-Ons For The Blind 295

TEXAS INSTRUMENTS
TI Forum 147

HEATH/ZENITH
The Next Greybox Generation 151
SigmaSoft's CAD-89 151

SPOTLIGHT ON BUSINESS
The Legal Professional A Fertile Ground For Computers 155

MISCELLANEOUS
Sony And The Multiscan Gang: Monitors For The Future 159
X Window: Promising Windowing Standard 159
Cheapsquawk: Low-Cost Sound For Surplus "Mute" Monitors 160
Xerox Desktop Publishing Has Seven Systems Offerings 288
Compaq Computer Corporation Introduces Lighter, Smaller Full-Function Portable Computer 292
Atari Releases Photos of New Computers 307
Digital Research Inc. Introduces New Operating System For 80386-Based PCs 307
SumTime: A Powerful Time Manager and Scheduler 318
ICCP Announces 1987 Exams For Computer Professionals 321
AIC *Plus* To Go On-Line 321

OTHER
Coming Events 320
Users Groups ..... 332, 334, 336, 338 & 339
Bulletin Board ..... 322, 323, 324, 326, 328 & 330
Showcase 340, 342, 344, 346, 348, 350
Classifieds 352
Advertiser Index 4
Service Directory 378

COMPUTER SHOPPER®

Audit Bureau of Circulations membership applied for.

Computer Shopper **(ISSN 0886-0556) is published monthly for $21 per year by *Computer Shopper of Titusville, Inc.*, 407 S. Washington Ave., Titusville, FL 32796. Second-class postage paid at Titusville, FL and additional mailing offices.**
POSTMASTER: Send address changes to *COMPUTER SHOPPER*, P.O. Box 1419, Titusville, FL 32781.

Entire Contents Copyright © 1986 *Computer Shopper of Titusville, Inc.* —All rights reserved.

Subscription $21 Per Year (USA Only)
***COMPUTER SHOPPER*, Inc.**
407 S. Washington Ave., P.O. Box F
Titusville, FL 32781
(305) 269-3211
Source ID No.: TCS575
Compuserve No. 70275,1023

NOTICE TO ADVERTISERS

All advertising is subject to Publisher's approval and acceptance without recourse. The advertising herein that is typeset and/or designed by *Computer Shopper*, is the property of *Computer Shopper* and not that of the advertiser. The advertiser has purchased the right of reproduction in *Computer Shopper* only and does not have the right to reproduce the ad in any other place of publication. **We are not responsible for mistakes, misprints, or typographical errors and will not issue credits of any kind for such errors.** Advertisers and their agencies assume responsibility for the condition of the contents of advertising printed, and agree to indemnify the Publisher of *Computer Shopper* for any and all claims and expenses arising therefrom. *Computer Shopper* standards suggest to the advertiser that statements concerning shipping and handling charges, warranties and money/or no money back guaranties should be stated in all forms of advertising within *Computer Shopper*. The opinions expressed in the articles and ads are those of the writer and not necessarily those of *Computer Shopper*.

PUBLISHER'S POSITION ON HONESTY

We guarantee that all ads meeting our ad policies will be published. There is no "skimming" bargains off the top or letting anyone preview the ads of the next issue.

We have expended much time, effort and money with the ultimate goal of providing subscribers with a nationwide ad service for the lowest possible cost. In reciprocation, we would expect you to conduct your transactions fairly, honestly and in the same manner as you would expect others to deal with you.

There are a few people who appear unable to go through life without having a hassle over every deal they encounter. If you are this type of person, we prefer not to handle your ad business.

We will not knowingly publish a classified or commercial ad of which we have information that the individual or firm is not reputable and fair. If you answer an ad within *Computer Shopper* and encounter dishonest treatment, please let us know. If there are several legitimate complaints, we will refuse future advertising and if a subscriber, cancel their subscription.

Richard Govatski
Group Publisher

FOUNDER—Glenn E. Patch
GROUP PUBLISHER—Richard Govatski
PUBLISHER—Stan Veit
EDITORIAL
EDITOR-IN-CHIEF—Stan Veit
ASSOCIATE EDITOR
John Anderson
TECHNICAL EDITOR
Ted Drude
ASSISTANT EDITOR—Annda Kinsey
EDITORIAL ASSISTANT
Rose Restina
CONTRIBUTORS
Dr. Ron Albright, Richard Fujita
Lon Andrews, Jeff Brenner
Philip Chien, Don Lancaster
Cheryl Peterson, Mark Fendrick
Wynn Rostek, Ervin Bobo
Ronald Bobo, Andrew Fried
Wayne Yacco, Johnathan Zittrain
Barbara & John McMullen
Hal Nieburg
PRODUCTION
PRODUCTION DIRECTOR
Bernard Monroe
ADVERTISING PRODUCTION COORDINATOR
Sam Batts
ADVERTISING PRODUCTION ASST.
Nancy Altman
ASSISTANT PRODUCTION MANAGER
Debi Hartling, Mary McDonel
COPY EDITING
Charlotte Barbour, Bonita Paulk
COMPOSITORS
Patti Dixon, Pamela Dowdy
Vicky Hamilton, Candace Cardonia
ART DIRECTOR
Debi Hartling
ASSOCIATE DESIGNER
Wendy LaPinta
ADVERTISING SALES
ADVERTISING SALES DIRECTOR
Mark Evans
ACCOUNT EXECUTIVES
New England—Theresa Brown
Mid-Atlantic—Ken Wood
Southeast—Theresa Brown
Midwest—Larry Morrissette
Southwest—Bob Harper
Southern Calif.—LaNeal Daniel
Northwest—Gary Liput
WEST COAST NATIONAL ACCOUNT REPRESENTATIVE
Joe Mesics
Jem Associates
1905 Pierce St.
San Francisco CA 94115
(415) 563-3230
CIRCULATION
DIRECTOR—Rosemary E. White
DIRECTOR, SINGLE COPY SALES
James Gerth
ADMINISTRATIVE ASSISTANT
Lori Conway
FULFILLMENT COORDINATOR
Pat Irvine
CREATIVE SERVICES
MARKETING MANAGER
Kevin Mulligan
PROMOTIONS MANAGER
Kim Mills-Price
RESEARCH ANALYST
Brett Alexander
COMMUNICATION DESIGNER
Richard Murray

COVER PHOTO
Rick Bostick

THINGS TO COME

- *Computer Shopper's* Clone Validation Suite
- Compaq's New Portable—A Close-Up Look
- CLARION—A Hands-On Review

Patch Communications

CORPORATE STAFF

CHAIRMAN—Glenn E. Patch
PRESIDENT/C. E. O.
Michael F. Billings
VICE PRESIDENT/C. F. O.
Donald F. Welk
VICE PRESIDENT/BUSINESS DEVELOPMENT—Linda Rew
ASSISTANT TO PRESIDENT
Sherry Allison
COMPUTER OPERATIONS DIRECTOR
Dan Wade
DIRECTOR HUMAN RESOURCES
Kathy Miller
ACCOUNTING MANAGER
Beverly Ryan
CREDIT MANAGER
Eileen McCullough
CUSTOMER SERVICE REPRESENTATIVE
Sherry Allison

ATARI

Applying The Atari

by Jeff Brenner

This month's program is a utility designed especially for the BASIC programmers. Automatic Voice Silencer essentially adds an extra parameter to the Atari BASIC SOUND command, allowing you to specify a length of time for the given sound. We've also got reader mail and newsletters.

Reader Mail

Dear Jeff,

Recently I have decided to undertake a hardware project. As I have some questions, I thought that perhaps you can answer a few.

I have an Atari 800XL with BASIC Rev. B but I would like to change the BASIC Rev. B ROM chip on the motherboard to the ROM chip with BASIC Rev. C. Since I already have BASIC Rev. C in cartridge form, I can remove the chip from the small PC board by desoldering it. I then hope to remove the ROM chip on the motherboard that has BASIC Rev. B and put in its place the ROM chip with BASIC Rev. C. I realize that I may have to desolder the chip on the motherboard, but I am confident that I can perform the desoldering because I have done delicate and careful desoldering before. I have two questions concerning this project. The first question is whether ROM chip Rev. B is electrically compatible with ROM chip Rev. C? The second question is whether ROM chip Rev. B is pin compatible with ROM chip Rev. C?

J. V. Martinez
Los Angeles, California

Since we can plug either a B ROM BASIC or a C ROM BASIC into the cartridge port, it is safe to assume that the two are electrically **and** pin compatible (the ROM chips are wired or plugged into a standard cartridge shell). The only difference between the two is the data contained within the ROM chips, and this data merely serves as a program that the Atari will execute.

I should warn you, however, of the possibility of damaging either a ROM chip or the circuit board, particularly since you mention desoldering—so **do** exercise caution if you intend to proceed with your project. Of course, if the chips are socketed (as might be the case with older cartridges and Ataris) you're in luck since you should be able to pull the old chips out and replace them without the need to desolder.

Personally, I probably would not bother replacing the ROM chips and would instead leave the C ROM BASIC in the cartridge port (the Atari would then run the C ROM BASIC instead of the B ROM after powerup), but if you are intent upon making the chip swap, then good luck.

Dear Jeff,

I'm very curious to see how the checksums used with Program Perfect are generated. I didn't spend an incredible amount of time trying to figure it out, but to no avail anyway.

Larry White
Mooresville, Indiana

The Program Perfect codes are generated by first summing up all of the characters on a line. The resultant sum is then converted into a two character code based on the letters of the alphabet. A third character tells Program Perfect where the next line is for the automatic line numbering feature. (If the third character is a "J," as it often is, this means that the following line is 10 away from the previous, since J is the tenth character of the alphabet). The last line of a program contains a "Z" as the third character in the code. When you later enter these codes, the "Z" tells Program Perfect that the entire program has been entered.

One question I've received a number of times is, "Is Program Perfect really perfect?" and my answer is "No." No proofreading program could be absolutely perfect unless the codes it used were as long as each program line itself. With a three character code (actually a two character code and a line numbering code) there is always a possibility that an incorrectly typed line can contain two errors that "cancel" each other out and result in the correct code. With Program Perfect, this can occur if two characters are transposed, since character position is not taken into consideration when the code is being generated. In most instances, the result would be a syntax error (i.e. **PIRNT** instead of **PRINT** would be caught by the program for the syntax error rather than for a code mismatch). However, characters transposed within a string or on data lines will not be spotted by Program Perfect, so these should be entered carefully. Ideally, Program Perfect would combine a position code into the checksum so as to catch these transposition errors, but changing the coding procedure at this point would only cause mass confusion among those using Program Perfect (particularly among newsstand buyers who might miss the issue containing the update).

You'll find two programs that will take a BASIC program LISTed to disk (i.e. LIST "D:TEMP" and print it along with Program Perfect codes. The listing under "Program Perfect Coder" will work with any printer, while the "Program Perfect Coder 2" listing will work with Epson/Gemini/Star models. The second version prints a centered title and automatically advances the page when necessary. With minor modifications, character position can be added into the coding (for example, line 70 could contain T = T + ASC(D$(I,I)) + I) or the coding procedure could be modified to utilize numbers in addition to letters, or to incorporate an entirely new method of coding. Remember, though, that any changes made to the coder must be made to the corresponding "Program Perfect" (or other typing checker) so the two will remain compatible.

April Fools'

Undoubtedly, many of you have been patiently waiting for the living-cell, biologically interfaced 85-megabyte organic memory expansion project with intelligent decision-making capabilities. While I might have made some of the folks at IBM a bit nervous about what I was going to reveal, we don't have room this month for the project. Do you believe that? Maybe next April.

Newsletters

From the Lake County Illinois Atari Computer Enthusiasts comes the L.C.A.C.E. Newsletter ($15/yr, P.O. Box 8788, Waukegan, IL 60079-8788). The January issue contained a number of reviews, news from the fall Comdex, and a section for the ST. Does your group publish a newsletter that we haven't seen yet? If so, send it in!

Automatic Voice Silencer

The Atari computer has the ability to generate sound independent of the actions of its 6502 processor chip, since sound generation on the Atari is controlled by separate circuitry. A sound that is turned on will remain sounding until it is forced off with another sound command (i.e. SOUND 1,0,0,0 silences voice 1). This independent sound generation offers many advantages since the computer can perform other operations while a sound is on. For example, the action in a game does not need to be interrupted to continously generate a sound, thereby allowing faster and smoother game play. Or a music program can sound notes while it is reading data for the next notes to be sounded, or loading data to be drawn on the screen.

Despite the advantages that the Atari's sound generation offers to the programmer, the fact that the sound must be turned off by the program can be a serious inconvenience. A BASIC programmer, for example, must somehow keep track of the length of time a sound is being produced, and then turn the sound off after the desired duration. This often involves placing a FOR-NEXT loop after the sound statement (i.e. SOUND 1,25,10,8:FOR I = 1 TO 30:NEXT I:SOUND 1,0,0,0). But by doing this, the programmer is defeating the purpose of the Atari's independent sound registers, since the computer is effectively tied up until the loop has ended. The computer could be doing other tasks while this sound is being made, but then the programmer can have a difficult time finding a place in the program where the sound could be turned off after a given amount of time.

The subroutine in this article eliminates this problem by allowing you to specify an amount of time a sound is to continue sounding. After the specified amount of time has elapsed, a machine language routine automatically turns the sound off.

How It Works

The subroutine in lines 30000 through 31070 stores a machine language program in memory. This machine language program is set up as part of the Atari's Vertical Blank routine. This allows the program to constantly monitor and control the computer without interferring with the program the computer is running. Thus, it is able to turn any of the Atari's sound registers off at any time. The program uses memory location 204,205,206 and 207 ($CC, $CD, $CE and $CF) as counters to determine when to turnoff a sound register. Location 204 is the timer for sound register #0; location 205 is for sound register #1; location 206 is for sound register #2; and location 207 is for sound register #3. Another timer, at location 203 ($CB) is used to count at a rate of ten times per second. Each time locations 204 through 207 are changed, location 203 should be reset to zero (with a POKE 203,0).

Using The Automatic Voice Silencer

Before using the Automatic Voice Silencer, your BASIC program must first GOSUB 30000. This subroutine will load the machine language routine into the computer's memory. It is necessary to GOSUB to this line only once. From then on, locations 204, 205, 206 and 207 can be used to time sounds. Before any sound statement that you wish to time, do the following:

1) Determine the time period for which you want the sound to continue. The time must be specified in tenths of a second. You can use any time from one to 255 tenths of a second. If you wish to specify eight seconds, for example, you would use eight times ten, or 80 tenths of a second.

2) POKE location 204, 205, 206 or 207 with this time. The voice number producing the sound determines the location which must be poked (thereby allowing each voice to be controlled independently.)

SOUND STATEMENT	USE LOCATION
SOUND 0,0,0,0	204 (POKE 204,T)
SOUND 1,0,0,0	205 (POKE 205,T)
SOUND 2,0,0,0	206 (POKE 206,T)
SOUND 3,0,0,0	207 (POKE 207,T)

3) POKE location 203 with a zero. This resets the program's main timer (POKE 203,0).

After these steps, your sound statement(s) would follow.

Entering The Program

Type in the listing under the "Automatic Voice Silencer" heading using Program Perfect and LIST it to disk (i.e. LIST "D:VOICE") or cassette so you can ENTER it later to merge it with your BASIC programs.

Now enter the Demonstration program (lines 10 through

continued on page 258

Applying The Atari continued from page 115

120) without erasing the subroutine. When you RUN this short program, it will initiate the Automatic Voice Silencer and will then sound four notes simultaneously. Lines 70 through 100 of this program give each voice a different duration from two seconds to eight seconds. After the program ends, it will print "LISTEN..." and you should be able to hear each voice being turned off one after the other. Note that the demo program itself does not shut off the voices, but merely specifies the duration.

By adding specified time durations to independent sound generation, the Automatic Voice Silencer creates a more versatile sound function. Use it to free up the time between generating a sound and turning it off so you can use the Atari's sound registers as they were designed to be used.

Coming Up

For those of you who can't get enough heavy typing to do, a bank account recordkeeping and checkwriting program is in the works. Stay tuned...

Readers' questions, comments and contributions are welcome. Please enclose a self-addressed, stamped envelope (SASE) for a personal reply. Due to volume of letters received; only a selected number of personal responses can be given each month.

A diskette containing the programs listed in this month's column is available from the author for $7, postpaid. Please make your check payable to "Jeff Brenner" and specify your disk drive model.

"Program Perfect" is a utility used to check for typing errors while entering programs from this column. Readers can send $5 for a diskette or a SASE for a listing of this program.

Address all correspondence to Jeff Brenner, "Applying The Atari 4/87" c/o *Computer Shopper*, P.O. Box F, Titusville, FL 32781-9990. ●

PROGRAM PERFECT CODER

```
MUE 10 DIM D$(100),NAME$(100):D$="D:":PRINT "NAME OF FILE: ";:INPUT #16;NAME$
KME 15 INPUT #16;NAME$:D$(LEN(D$)+1)=NAME$
BSJ 20 PRINT "PRESS SPACE BAR FOR PAGE BREAK"
WAJ 30 OPEN #1,4,0,D$:T=0:INPUT #1,D$
ORJ 40 IF D$(LEN(D$))=" " THEN D$=D$(1,LEN(D$)-1)
FUJ 50 TRAP 60:INPUT #1,NAME$:NL=VAL(NAME$)-VAL(D$):GOTO 70
BYJ 60 NL=26:E=1
AOJ 70 T=NL:FOR I=1 TO LEN(D$):T=T+ASC(D$(I,I)):IF T>650 THEN T=T-650
BHJ 80 NEXT I:C1=INT(T/26):C2=T-C1*26
FSJ 90 PRINT #3;CHR$(C1+65);CHR$(C2+65);CHR$(64+NL);
IHJ 100 PRINT #3;D$
SGJ 110 IF E=1 THEN END
HGJ 120 D$=NAME$:IF PEEK(764)<>33 THEN 40
SZJ 130 POKE 764,255:PRINT "PRESS RETURN WHEN READY"
NAJ 140 IF PEEK(764)<>12 THEN 140
ADZ 150 GOTO 40
```

PROGRAM PERFECT CODER 2

```
KAE 10 OPEN #3,8,0,"P:":DIM E$(1):E$=CHR$(27)
QHE 15 PRINT #3:PRINT #3:PRINT #3:PRINT #3
BJJ 20 PRINT #3;E$;"@";E$;"4";E$;"E";E$;"N";CHR$(10);
MXE 30 DIM N$(40):PRINT "TITLE";:INPUT N$
UDE 35 PRINT #3;E$;"W";CHR$(1);E$;"4"
IVJ 40 FOR I=1 TO (40-LEN(N$))/2:PRINT #3;" ";:NEXT I
BWJ 50 PRINT #3;N$;E$;"W";CHR$(0);E$;"5"
WKJ 60 PRINT #3:PRINT #3
YHJ 70 DIM D$(100),NAME$(100):D$="D:":PRINT "NAME OF FILE: ";
RXJ 80 INPUT #16;NAME$:D$(LEN(D$)+1)=NAME$:PRINT #3;E$;"A";CHR$(14);
WGJ 90 OPEN #1,4,0,D$:T=0:INPUT #1,D$
SKJ 100 IF D$(LEN(D$))=" " THEN D$=D$(1,LEN(D$)-1)
KZJ 110 TRAP 120:INPUT #1,NAME$:NL=VAL(NAME$)-VAL(D$):GOTO 130
DRJ 120 NL=26:E=1
CHJ 130 T=NL:FOR I=1 TO LEN(D$):T=T+ASC(D$(I,I)):IF T>650 THEN T=T-650
QAE 140 NEXT I:PRINT #3;E$;"F";:C1=INT(T/26):C2=T-C1*26
VGE 145 PRINT #3;CHR$(C1+65);CHR$(C2+65);CHR$(64+NL);E$;"E";
DLJ 150 PRINT #3;" ";E$;"E";D$
SLJ 160 IF E=1 THEN END
JEJ 170 D$=NAME$:IF PEEK(764)<>33 THEN 100
TEJ 180 POKE 764,255:PRINT "PRESS RETURN WHEN READY"
NKJ 190 IF PEEK(764)<>12 THEN 190
BSZ 200 GOTO 100
```

AUTOMATIC VOICE SILENCER

```
LWJ 20000 REM AUTOMATIC VOICE SILENCER
KLJ 20010 REM BY JEFF BRENNER
DEJ 20020 REM COMPUTER SHOPPER, 4/87
LTJ 20030 T=0:I=0:RESTORE 20140
BGJ 20040 READ N
APJ 20050 IF N=-1 THEN 20090
QEJ 20060 POKE 1536+I,N
QTJ 20070 I=I+1:T=T+I+1
IZJ 20080 GOTO 20040
EYJ 20090 IF T=1325 THEN 20120
PUJ 20100 PRINT "ERROR - CHECK DATA LINES"
XOJ 20110 STOP
LXJ 20120 I=USR(1536)
EOJ 20130 RETURN
OXJ 20140 DATA 104,162,6,160,11,169,7
TDJ 20150 DATA 32,92,228,96,230,203,165
OSJ 20160 DATA 203,201,6,208,23,169,0
QRJ 20170 DATA 133,203,160,6,162,3,181
UKJ 20180 DATA 204,240,6,214,204,181,204
SUJ 20190 DATA 240,8,136,136,202,16,241
QZJ 20200 DATA 76,98,228,153,1,210,240
JYZ 20210 DATA 243,-1
```

VOICE SILENCER DEMONSTRATION

```
AQJ 10 REM DEMONSTRATION PROGRAM
FSJ 20 GOSUB 20000
PHJ 30 SOUND 0,121,10,8
POJ 40 SOUND 1,162,10,8
PUJ 50 SOUND 2,193,10,8
PRJ 60 SOUND 3,242,10,8
XBJ 70 POKE 204,20:REM TWO SECONDS
ATJ 80 POKE 205,40:REM FOUR SECONDS
XDJ 90 POKE 206,60:REM SIX SECONDS
EPJ 100 POKE 207,80:REM EIGHT SECONDS
XWJ 110 POKE 203,0:PRINT "LISTEN..."
ULZ 120 STOP
```


IBM® XT COMPATIBLE

CAL COMM CENTRE
(818)300-8599 7 DAYS
523-C SO. ATLANTIC BL. MONTEREY PARK, CA. 91754

XT TURBO 640 $629.00
NO HIDDEN EXTRAS

• 640K TURBO M/B FCC	112
• 640K MEMORY ON BOARD	60
• MONO GRAPHIC/PRINTER	60
• 135W POWER SUPPLY	56
• TWO FUJITSU FLOPPY D/D	158
• FLOPPY D/C/CARD 2 DR.	27
• FLIP TOP CASE	27
• SAMSUNG 12" AMB/MONITOR	79
• AT STYLE KEYBOARD	50

XT COLOR 640 $879.00
QUALITY SYSTEM

• 640K XT M/B FCC APPV.	99
• 640K MEMORY ON BOARD	60
• COLOR DISPLAY CARD	60
• 135W POWER SUPPLY	56
• TWO FUJ. FLOPPY D/D	160
• MULTI I/O W/FDCC DTK	79
• FLIP TOP CASE	27
• 14" N'GLARE COLOR MO.	285
• AT/XT SWITCH KEYBOARD	53

HARD DISK 640 $1039.00
FORMATTED

• 640K TURBO M/B FCC	112
• 640K MEMORY ON BOARD	60
• MONO GRAPHIC/PRINTER	60
• 150W POWER SUPPLY	62
• ONE FUJ. FLOPPY D/D	80
• 20MB SEAGATE W/C'LER	415
• MULTI I/O W/FDCC DTK	79
• FLIP TOP CASE	27
• SAMSUNG 12" AMB/MONITOR	79
• 5151/5152 KEYBOARD	65

OPTIONS:

• PC/XT USERS GUIDE	12.75
• MS-DOS 3.1 US/GUIDE & DISK	25
• PRINTER CARD	16
• PRINTER CABLE XT	10
• MULTI I/O FCC APPROV.	83
• MONO GRAPHIC FCC APPROV.	68
• 30MB HARD DISK W/C'LER	CALL
• 14" COLOR/AMB/GRN S'CHABLE	295
• INTERNAL MODEM 1200BPS	120

※ DEALERS WELCOMED
※ QUANTITY DISCOUNT
※ 90 DAYS WARRANTY
※ CHOICE OF ANY KIT
※ ASSEMBLE & TEST $30
※ COD OK W/$200 DEP.

ATARI ST

GFA BASIC for The ST

by John J. Anderson

I'm one of those people who think that BASIC has gotten a bum rap. It is, of course, the programming language I learned first, and arguably best (simply by exposure), and I am by no means an assembly whiz or C protege. For my kind of dabbling, BASIC has long been a comfortable place to be.

Unfortunately, it has never offered the kinds of capabilities to be found with other programming languages. It has been slow, weak, unable to hook into the strongest facets of a machine—programs written in BASIC have a tendency to telegraph themselves in the worst ways. Know what I mean? "This is so bad, it must be written in BASIC," you may occasionally find yourself saying to yourself.

Even worse, and largely responsible for the notorious reputation (or perhaps I should say "lack of") displayed by the BASIC language has been the nearly barbaric programming conventions it seemingly encourages. Certainly sloppy programming and damnable coding habits are ultimately the responsibility of the programmer himself, and no one else. But BASIC undeniably tempts the holiest of the holy—with its line numbers, its untaggable variables, and its lowly desperado of a command, GOTO. In a showdown, GOTO will plug you first every time.

I call myself a BASIC fan, yet I'll freely admit that if called upon to decipher some code, I'd rather look at Pascal or Logo hands down. Maybe even LISP, if I'm feeling frisky.

Why is this? Because languages such as Pascal are "structured," that is, designed to encapsulate code routines in separate modules, callable by some logical name, and much easier to understand in a much shorter period of time. There are no line numbers (at least that the programmer need concern himself with) and no GOTOs. Rather there are procedures and procedure calls. When reading someone else's code, you can see paths quickly. When writing your own code, you don't need to leave a trail of breadcrumbs to retrace your steps. Neat.

It should be noted that there has been somewhat of a backlash against Pascal in recent months, and I do understand it. Pascal forces you to declare yourself in Pascal's own terms, to put all your cards on the table right at the beginning, as it were, and to think the way Nikolas Wirth thinks, if you intend to get anything substantive done. In its desire to keep you clean and coding on the right track, it constricts you, and this may inhibit inspiration. David Small, longtime Atarian whose latest venture is the Macintosh Emulator Magic Sac, has called Pascal "a Fascist language." A tad overstated, perhaps, but I can certainly see where he is coming from. In its own way Pascal is almost as bad as Cobol, in respect to the requirements of even the simplest of jobs. Sometimes you need to cross a ravine and what you want is a strong plank—you don't need a suspension bridge every time.

But let me remind you, I am not a programmer, I am a writer who programs as a hobby. And I am one of those people who think that BASIC has gotten a bum rap.

Billing itself as "the full access" BASIC programming language for the Atari ST series, GFA BASIC, from GFA Systemtechnik in Germany, fulfills that promise better than any other ST BASIC I have seen to date. It has hooks into most of the powerful features of the ST, and is quite fast (more a reflection of the ST's 68000 processor than anything else). It is possible to write a program in GFA BASIC that in no way telegraphs the fact that it is not written in C, or Forth, or assembly. Huzzah. It lists for $79.95. Double huzzah.

What's more, GFA BASIC is a structured BASIC. It sports no line numbers, and though it does feature a GOTO statement, you can hold it in abeyance, rather like the Hyperspace button on an arcade game. If your back is to the wall, and there's no other way out, your old buddy GOTO will shoot you out of there in no time. But he leaves your code a messy place. Better bring a mop.

In addition to the fact that GFA BASIC supports procedures, it allows exactly and only one command per line. This may seem like a throwback, and I can tell you that it does take some getting used to. But if memory is not a problem (and on the ST it is not, or at least should not be a problem), the approach results in code that is much easier to follow. Sure, your listings are a lot longer than they would be with multiple command lines. But at the same time, they are much less dense. It is less likely that you will miss something salient as your eye scans a chunk of code. And what may have seemed like a constraint at first quickly evolves into a programmer's aid, once you ease into it.

The editor within GFA BASIC has its nuances, but on the whole is a pleasure to use. It is what I call a "real-time" editor: that is to say you can change the code on a line, click RUN, and watch the modified code execute. There's no pressing of RETURN to enter a line, or a Byzantine windowed entry system as presented by Microsoft BASIC for the Mac. Simply make your alterations and go. Insert, delete, block move and copy, search and replace, all work exhilaratingly well. True, marking blocks and manipulating them is a two-step process, and can seem a bit awkward. Paging does not work window-style, either, but with mouse clicks on the menu bar, which is a little distracting, at least to start with. But the editor is smooth, robust, and powerful. It includes a large set of cursor movement commands. It allows you to flip between the editor and display screens, to see the results of your code upon it. It allows you to toggle between insert- and overwrite-style input. It even features a pseudo-direct mode, where commands can be entered command-interpreter style. TRON and TROFF commands there aid in debugging code.

Error messages in GFA BASIC are always full-text,

continued on page 120

Hacking The ST

by Wynn Rostek

This month, I'll tell you a little about the way the 68000 CPU is built, at least as far as the programmer views such things. Let's take a look at a typical 68000 assembly language instruction.

```
load: move.w #scrrez,d0 * find resolution
```

This line of assembly language code is known as a statement. The first part of the statement, for those of you new to assembly language programming is called a label. Labels allow you to refer to things symbolically. If you wanted to run the subroutine that started with the above line, you would use:

```
jsr     load
```

The assembler will figure out the address of the instruction that started with "load" and would substitute that value in the jsr load instruction. This has to happen because all the CPU understands is memory addresses, it has no idea what "load" is, it insists on having an address.

The label portion of a statement is optional. If you never need to refer to the rest of the statement that appears on a line, there is no need to include a label as the first element.

The second part of the statement is the mnemonic. This is either a instruction that the CPU can perform like move, or add, or it could be a directive to the assembler program like page or title.

You should have noticed (if you have been reading this column, or have seen 68000 assembly language elsewhere) that the mnemonics for the 68000 often end with a .b or a .w or a .l. These endings specify the size of the data object we are working on.

Any instruction ending in .b refers to a byte or an 8 bit quantity. A .w ending means we are working with a word or 16 bit quantity. The .l ending specifies a long or 32 bit quantity.

After the mnemonic comes the operands. Operands are arguments used by the assembler to build the complete machine language instruction. If we encounter the mnemonic move.b, we already know that we are going to move 8 bits, but the question is where do the 8 bits come from and where do we move them to?

In the case of the move instruction, the operands tell us where the data comes from (the source) and where the data ends up (the destination).

The operands are separated by a comma. Following the operands is a comment. The rules for where comments start vary from assembler to assembler. Some assemblers allow anything following the operands to be a comment. Others require a delimiter to notify the assembler that a comment is starting.

The assembler that comes with the developers package uses the "*" character to denote the start of a comment. Almost all assemblers agree that comments stop at the end of the line.

You will notice that many of the instructions in the 68000 use 2 operands data, they also can use 2 different addressing modes in a single instruction. In fact, most of the 68000 instructions use two operands and can use two different addressing modes. Because most instructions have 2 different addresses in them, the 68000 is called a two address machine.

As an aside, the 2 address machine is the most popular now in terms of number of CPUs using it, but there are 3 address machines, 1 address machines and even 0 address machines.

While there are not many 3 address machines around, a compiler often generates 3 address instructions (called quads because there are three addresses and one operation) and then translates the three address code to actual instructions for the target CPU.

A 0 address machine is a stack machine (perfect for running Forth). Almost all operations push or pop values off of the stack to get the work done.

Even a 0 address machine needs to have a few single address instructions. Whether a machine is 2 address or 1 address or whatever depends on the most common instructions in it's instruction set. If most of the instructions are 2 address instructions, then it is a 2 address machine even if it has some 1 address or 0 address instructions.

continued on page 120

GFA BASIC
continued from page 119

and usually point you directly to the problem. The interpreter is quite "smart," and can often accept a line that has been entered incorrectly, fixing the mistake as it goes (as long as the mistake is a minor one). As an example, a line you enter as ?at (2,2) "Test" will click in as Print At (2,2); "Test" and execute correctly. Insertion of the semicolon was especially impressive to me. Even when your code results in a crash, BASIC will tell you how many bombs your ST has returned and what they might mean to tell you.

The function keys on the ST are fully enabled by GFA BASIC, and can shortcut point and click commands to load, save, save in ASCII format, merge, list files to a printer, flip between screens, go to direct mode, test program loops, run a program, or quit.

GFA BASIC's command set is extremely complete, and we don't have the room here to list them all. Suffice to say that you won't find it lacking any command you want to use, and in fact will discover it is replete with commands you didn't know you needed, but will happily learn to use. Among these are commands for graphics, sound, menus, windows, mouse, machine code calls, drive control, time and date, directories, file manipulation, various input formats, a wide range of loop formats, string handling (ala Microsoft), error trapping, I/O channels, peeks and pokes, screen manipulation, READ-DATA manipulation, pointers, and traces. Special functions include VDI and AES calls, including window control through GEM, the capability to pass on variables and arrays indirectly, a raster copier command, fast movement of memory blocks, monitor calls, and true and false conditions.

The EXEC command allows you to load and execute a non-BASIC ST application and run it from within a GFA BASIC program. The C: command calls and runs a routine written and compiled in C.

Variables can be in the form of single letters or words, and the full word is always used by GFA BASIC to distinguish between variables. If a value is assigned with the LET command (optional, of course) then even command names are allowed as variable names—you can say LET LET = 17, and it will happen. Real, integer, Boolean, and string variables are supported. Significantly too, local variables are also supported.

Included on the disk with GFA BASIC is a program that strips line numbers from ST-BASIC programs for conversion to GFA BASIC. The program helps show you where you'll need to do some custom tailoring. Also on the program disk is a run-only version of GFA BASIC, which is a quasi-compiler. It allows you to run GFA BASIC programs as stand-alone applications, and GFA Systemtechnik invites you to copy the run-only module at your pleasure. My informal tests showed no appreciable speed difference between the actual program and the run-time compiler—but this was only because both were so fast.

So it becomes clear that GFA BASIC is an excellent program, and an excellent programming language. It brings BASIC to the ST in a manner truly worthy of the machine. In fact, it could be said that GFA BASIC is the only BASIC programming language yet released that presents a good, strong argument for programming an ST in BASIC in the first place.

That is not to say that it is entirely without its disappointments. The documentation comes from the German, and suffers badly from "translationitis," wherein quotes are called "inverted commas" and the like. In places it is verbose about a simple point, in other places terse about complex ones. It transposes bars for slashes, ohs for zeros, the words TEST and TEXT, and then there are the simple typos in program code. I fear that the documentation accompanying GFA BASIC will hinder its deserved acceptance as the de facto BASIC language for the Atari ST, as it camouflages the elegance and ease of use of the package.

Then there are the problems of hooking into GEM. These cannot fairly be placed at the feet of the people at GFA Systemtechnik—but in their efforts to bring the power of the ST to the commonfolk, they get caught in the middle. Simply opening and updating a movable window on the ST is an exercise in the complexities of GEM (see example in accompanying program). But perhaps I am revealing my own deficiencies in this comment, rather than those of the systems I describe.

Listing 1 is a program I worked up that demonstrates the capabilities of what I believe to be the most interesting commands of GFA BASIC. The goal is to demonstrate some of those capabilities, and to provide a tutorial for new users. This listing is available online through Compuserve and Delphi, along with the run-time package. By downloading the files you can have a look at GFA BASIC in the comfort of your own home, and perhaps get a hint of whether or not it is something you would like to purchase. I have annotated the program here (note that you won't be able to list the program with the run-time package).

What the program does is throw a menubar on the screen, offering selections for Intro, Windows, Mouse, and Graphics. You can pick and choose through these selections, and run through demonstrations of various commands. The following is a summary of the program itself:

The first module of the program defines the menubar and displays it on the screen. It initializes a text string (Bar$) which will contain all the elements of the menu, and uses READ and DATA statements

continued on page 259

Hacking The ST
continued from page 119

Well, the time has come to cover the 68000 addressing modes. The 68000 is a very powerful CPU, and one of the things that makes it so powerful is the instruction set, and a good portion of that power is a result of the different addressing modes it can use.

It might seem a little strange to devote a column to the addressing modes of the 68000, but you cannot really wring the last ounce of performance out of your ST unless you can write in 68000 assembly language, and to write in 68000 assembly language, you have to understand the addressing modes.

If you are saying to yourself right now, "What is an addressing mode?" you are not alone. Many people that have been casual programmers have never heard the term before.

When you are writing in assembly language, there is no compiler or interpreter to keep track of where things are kept in the machine's memory. You have to keep track of where everything is yourself, and somehow let the CPU know also.

There are basically two places that data can be stored as far as the 68000 is concerned, in a CPU register or somewhere in memory. If there are only two places to store data, why do we need addressing modes?

In programming, you often find yourself doing the same kind of job over and over again. As an example, imagine you are stepping through an array of data looking for a certain value. Your array of data is stored in the memory of your computer somewhere so all of the addresses you use when looking through the array access memory.

To illustrate the power of a good set of addressing modes, let's assume your array contains 10 integers and it starts at location 100. The value we are looking for is −1. Your first cut at code to search the array for this value might look something like the code in Table 1.

The move instruction should be obvious, the cmp instruction compares the two operands, and sets some flags based on the result of the comparison. If the two things being compared are equal, then the zero flag is set.

The beq instruction is a conditional branch. If the zero flag (one of the flags the cmp instruction can set) is set then the program jumps to the new address (found) but if the zero flag is not set, the CPU just executes the next instruction after the beq.

After looking at this for a short time, you will no doubt want to change it into a loop so that the job is done with far less code. Your first try at a loop might look something like the code in Table 2.

The first code segment uses immediate addressing, that is we tell the CPU where in memory to find the item by specifying the full address each time we get the value we want to look at. The first value comes from location 100, the second from location 102, the third from 104, etc.

In the second code segment we use address register indirect addressing so we can search the array with a short loop. Instead of specifying the memory address each time, we keep the address in register a0. After loading the value, we increment the contents of register a0 to point to the next value.

A third, and even better way to code the loop is presented in Table 3. This code segment uses address register indirect with post-increment addressing

continued on page 259

```
move.w    #100,d0     * get value 1
cmp.w     -1,d0       * see if we found it
beq       found       * got it...
move.w    #102,d0     * get value 2
cmp.w     -1,d0       * see if we found it
beq       found       * got it...
move.w    #104,d0     * get value 3
cmp.w     -1,d0       * see if we found it
beq       found       * got it...
move.w    #106,d0     * get value 4
cmp.w     -1,d0       * see if we found it
beq       found       * got it...
move.w    #108,d0     * get value 5
cmp.w     -1,d0       * see if we found it
beq       found       * got it...
move.w    #110,d0     * get value 6
cmp.w     -1,d0       * see if we found it
beq       found       * got it...
move.w    #112,d0     * get value 7
cmp.w     -1,d0       * see if we found it
beq       found       * got it...
move.w     #114,d0    * get value 8
cmp.w     -1,d0       * see if we found it
beq       found       * got it...
move.w     #116,d0    * get value 9
cmp.w     -1,d0       * see if we found it
beq       found       * got it...
move.w    #118,d0     * get value 10
cmp.w     -1,d0       * see if we found it
beq       found       * got it...
bra       gone        * it's not there
```

Table 1.
Searching an array, a primitive approach.

```
top:
move.l    #100,a0     * start of array
move.w    (a0),d0     * get a value
cmp.w     -1,d0       * see if we found it
beq       found       * got it...
addq.l    #2,a0       * next array entry
cmp.l     #120,a0     * array all done?
beq       gone        * end of array?
bra       top         * more to do
```

Table 2.
A loop to search an array.

```
top:
move.l    #100,a0     * start of array
move.w    (a0+,d0)    * get a value
cmp.l     #120,a0     * array all done?
beq       gone        * end of array?
cmp.w     #-1,d0      * see if we found it
beq       found       * got it...
bra       top         * more to do
```

Table 3.
A shorter loop to search the array.

Hacking The ST
continued from page 120

for fetching the value from the array. With address register indirect with post-increment addressing, the CPU automatically adds 2 to the contents of register a0 after it has performed the move.

By using the post-increment mode, we have managed to eliminate another instruction from our loop, speeding up our search even more. As you can see, the addressing mode used affects the efficiency of your code quite a bit.

I hope this short series of examples has shown you the importance of knowing the addressing modes that your CPU has. Knowing all the addressing modes your CPU uses can help you generate the most efficient code possible, while forgetting, or not knowing an addressing mode exists, can cause you to use very poor code to get a job done.

(For the people still reading that know 68000 assembly language well, there are branching instructions that allow us to do the search in only a couple of instructions, but I don't want to muddy the waters now with new instructions.)

The time has come to cover the addressing modes used by the 68000 CPU. The first addressing mode we'll cover is data register direct. In data register direct addressing, your data is contained in one of the 68000's data registers. In the cmp.w # – 1,d0 instruction, we let the CPU know that one of the values being compared is in register d0. This is data register direct addressing.

The next addressing mode is address register direct. The externally visible difference between data register direct and address register direct is whether you use an address register or a data register. There are some differences in the way the data is handled and what is allowed.

In address register direct addressing you cannot use any of the byte length instructions. move.b 3,d0 is legal but move.b 3,a0 is not allowed because it is a byte operation.

Another difference shows up when working with word length operations. When you move a word (16 bits) into an address register, the sign bit (bit 15, remember we start numbering bits with 0) is "extended" into the upper 16 bits of the register. (The data and address registers of the 68000 are 32 bits long.)

Bit 15 is the sign bit of a 16 bit number. If the most significant bit (bit 15 in our case) is a 1, then the number is negative, if it is 0, the number

continued on page 260

GFA BASIC
continued from page 120

to construct the string. A demonstration program on the GFA BASIC disk uses a totally different method to create a menubar, and although it reads menu selections more efficiently, I prefer the method shown here. It uses less brute force, is easier to track, and easier to modify. The menubar is constructed so desk accessories may be present on the boot disk. You can disable them easily if you so choose.

The next module is the main program loop, which, as you'll see, is a mere four lines long. It simply scans the menubar, and waits for a point and click. When it gets one, it moves on to Procedure Menu.

Now we reach our first defined procedure, Menu. You could handle the same task in a more efficient manner, but not a clearer manner. Here we identify each point and click by its actual name, and refer it to a specific procedure named in a similar manner. Each is sandwiched into an IF-END-IF loop. Why do it this way? Well the best reason is that it makes adding and subtracting elements from your menubar as straightforward as possible. Let's say you decide to eliminate one selection under Graphics, but you'd like to add a whole new header between Mouse and Graphics, with six new selections in it. If you read the menu numerically, you'll have to go in and re-engineer the whole Menu procedure. By doing it this way, all you need to do is revise the data statements, enter your new selections to Procedure Menu, and then make sure you add procedures for each new selection. In a way, this program can actually serve as a "template" for menu creation, and that is the real reason it got written. Note also that the escape hatch is located here. A click on "quit" returns us to the GEM desktop.

Next we see the menu components themselves. The first procedure displays the "About this program" box on the Desk accessories menu. As you can see, constructing an alert box is

continued on page 266

PRICE CUTS

ARNOLD COMPANY

LOWER PRICES

NEW PHONE: 214-395-2922 FOR SUPER FAST SERVICE!

COMMANDER XT

XT System Foundation 640K Commander Mainboard 256K dram installed high grade system case 150 watt power supply all hardware & speaker manual.

Special $259

5¼" POWER CONNECTORS!

FOR ALL 5¼ DRIVES

1	10	50	100
$1.00 ea.	$ .95 ea.	$ .89 ea.	$ .79 ea

5¼ DRIVE POWER CABLES

5¼ Power Connector on 2 ft. cable $1.75

EXTENSION 5¼ Power Cable 2 ft with Male on one end, Female on other $2.99

"Y" KIT ADAPTS YOUR SINGLE DRIVE P S FOR 2 HALF-HEIGHTS Includes 2 male connectors 1 female connector and wire

1	10	25	50
$3.75 ea.	$3.50 ea.	$3.35 ea.	$3.10 ea.

8" DRIVE POWER CONNECTOR SETS Set of 2 Specify your drive

1	10	25	50
$2.75 ea.	$2.50 ea.	$2.25 ea.	$2.00 ea.

TI99/4A KEYBOARDS!

UPGRADE your *Timex/Sinclair*. We provide detailed documentation FREE with your keyboard! $5.99

NUMERIC KEYPAD

Unencoded keypads feature each key terminated in its own switch. High quality NEW keypads have a great feel! $8.95

DIGITAL DRIVE DIAGNOSTIC

Test and Align your own drives using your computer, WITHOUT a Scope or costly equipment. Easy to use and accurate! Tests Speed, Clamping, Alignment Azimuth, Hysteresis. For IBM PC compatible MS-DOS computers. Includes Dysan Alignment Disk, Program and Comprehensive Manual $99

TI TERMINALS

TI Silent 700 used Communications printer & doc ..... Sold Out

TI700 Manual Only ..... Each $10.95

TI used 911-VDT video display terminal $49.95

COMPUTER AC CORDS

INDUSTRY Standard 6 FT. CORD FOR IBM, etc.

Chassis Mt. Socket for Above ..... $.75

Cords	1	10
	$2.50 ea	$1.90 ea

NEW! 25 ft. AC Cord for IBM ..... 6.50 ea.

COMMODORE serial cable. USR pt conn to DB-25 ..... $5

BUDGET ENTRY SYSTEM

A Complete 1 Drive System of Economy

- Commander XT + board with 256K installed (up to 640K on board) (Famous Commander Bios)
- 150 Watt Hard Disk ready Power Supply!
- Extra Heavy Gauge System Case!
- Color/Monochrome Graphic Adapter!
- AT style keyboard with enlarged Return, Caps/number lock lights!
- Just bolt it together, connect cables and start computing!
- All boards fully assembled and Tested!
- Assembly instructions 10 day money back offer 90 day warranty!

Special $445

Plus $12.00 shipping each

THE EXECUTIVE SYSTEM

A 2 Drive System Loaded With Extras

$629

COMMANDER XT with 256K Memory Installed
Slide Case and 150 Watt Power Supply
AT Keyboard with Crisp Keystroke
DISK I/O: (Par, Ser, FDC, Clock/Cal, Game Port)
Dual Ultra Quiet reliable Half Height Drives
Color Graphic or TTL Display Card

TRS-80 MODEL III CPU BOARDS

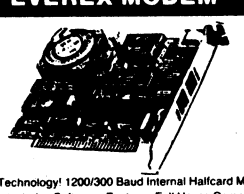
LIMITED SUPPLY!

Motherboards are surplus and come with Z-80, Character rom and other components. Each is complete except ROM/RAM. One punch hole in each board is easily repaired. Not junkers that have had all the socketed chips removed! Each is a rebuildable unit. W/Doc. $13.50

10 day no risk money back offer on each!

MODEL III PARTS BOARDS: A few other chips missing or more damaged and are offered for parts only at a reduced price ..... 3 for $15.00

EVEREX MODEM



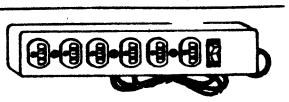
Latest Technology! 1200/300 Baud Internal Halfcard Modem for IBM, includes Software. Features Full Hayes Compatibility. Introductory Price $105

2400 Baud Modem $209

MASTER POWER PANEL

All Steel, Surge protecting master power control panel with separate illuminated switch for each of five peripherals! Attractive buff color, 10 amp. Surge/Spike protection panel fits conveniently between Computer & Monitor

Retail value $129 ..... Special! $47.95



PROTECTED POWER STRIP

Six switched plus illuminated switch and spike/surge protection at low cost $8.95

FLOPPY DRIVE CASE

New grey 2 piece horizontal case for ½ or ½ ht drives $8.95 ea or 2 for $16

uSDD 5¼" Diskettes W/Envelopes, Labels . 100 for $39

Smoke Top Locking 5¼" Disk Box $12.95

IBM PC/XT COMPATIBLES

COMMANDER XT

PRICE CUT!

20 MEGABYTE MASTER BUSINESS SYSTEM

OUR MOST POPULAR SYSTEM ON SALE

$929

COMMANDER XT with 256K Memory installed
Slide Case with 150 Watt Power Supply
AT Keyboard with Crisp Keystroke
DISK I/O: (Par, Ser, FDC, Clock/Cal, Game Port)
Ultra Quiet & Reliable Half Height Drive
20 Megabyte Half Ht. Hard Drive & Controler
Color Graphic or TTL Display Card

20 Megabyte hard Disk System includes Drive Controler, Cables Manual. 1 Year Warranty $369

BUILD YOUR OWN CLONE

PACKAGE:CASE/150 Wattps./Mainboard/256K	$259
COMMANDER XT 4.77 speed Mainboard W/Commander BIOS....The Most reliable in the industry! Up 640K on board. Maximum compatible! W/Ok	$109
8 Meg 4 layer Ultra Stable Turbo W/BIOS & OK	$139
DISK I/O (FDC, Par, Ser, Clk, Games /Cable	$85
Floppy Disk Controller only for 2 Dr W/Cables	$30
HARD DRIVE CONTROLER W/CABLES	$99
ADAPTEC 2070A RLL Hard Disk Ctrl Increase disk capacity by 50%	$139
Color/Mono Graphic Card (like IBM CGC)	$59
Monochrome Display Card	$65
Herc Compatible Mono Graph W/prtr port	$79
EVEREX ENHANCER EGA W/printer port	$239
EVEREX 1200 BAUD INTERNAL MODEM	$105
EVEREX 80286 SPEEDUP BOARD FOR XT	$LOW
Parellel printer adapter	$28
Serial Async. Comm. Adapter	$35
AT PAR/SERIAL ADAPTER	$59
AT FLOPPY/HARD DRIVE CONTROLER	$200
AT KEYBOARD	$89
AT CASE	$89
IBM 6 FT PAR PRINTER CABLE	$10
IBM 12 FT PRINTER PAR PRINTER CABLE	$17
IBM 6 FT SERIAL CABLE DB-25 M-F	$10
COPYIIPCOPTION BOARD	$89.99
PC/XT KEYBOARDS: OEM 5150 STYLE	$55
SELECTRIC XT KBD....GREAT FEEL	$59
AT STYLE KBD FOR XT...OUR FAVORITE	$59
5151 STYLE WITH SEPARATE CURSOR PAD	$69
CASE: HIGH GRADE SLIDE XT TYPE W/HARDWARE	$37
CASE: HIGH GRADE FLIPT TOP W/HARDWARE ($4 extra shipping per case)	$37
360 LATEST TECH. JAPAN DRIVES W/SYSTEM	$89.99
256K RAM (9 CHIPS)	$27
64K RAM (9 CHIPS)	$9
640K (18 256K, & 18 64K)	$72
12" COMPOSITE VIDEO MONITOR	$89
12" TTL MONOCHROME MONITOR	$99.99
HIGH RES RGB MONITORS	from $339
EGA MONITORS-MITSUBISHI-FROM	$499

IBM PC, XT is a registered trademark of International Business Machines Corp.

Dealer Inquiries Welcome

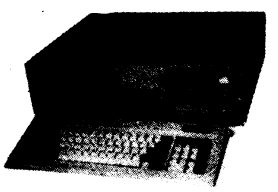
IBM PC/XT SOFTWARE ETC

COPYIIPC	$29.99
COPYIIPC OPTION BOARD	$89.99
MIRROR (Crosstalk Klone)	$45.99
Textra word Processor	$19.99
Digital drive Diagnost	$99
PRINTMASTER	$39.99

PUBLIC DOMAIN SOFTWARE $4.00/Disk

PC-FILE DATABASE	PC-CALC SPREADSHEET
PC-WRITE	RETAIL SALES (2DISKS)
CHURCH MANAGEMENT	FARM ECONOMICS
STOCK TRACKING	COMODOTIES TRACKING
PC DEMO SYSTEM	PRO-COMM COMM.
SIDEKICK KLONE	DOS-FILE UTILITIES

INTRODUCING OUR NEW TURBO AT COMPATIBLE



Dual Speed IBM AT Compatible

6/8 Mhz 80286 AT Compatible Mainboard
512K Ram Installed (up to 1 Meg on board)
Built in Clock/Calender w/battery backup
Fully Compatible BIOS Written in U.S.A.
200 Watt UL approved Power Supply
AT System Case with Keylock
5060 AT Tactile Feel Keyboard
1.2 MB or 360KB Floppy Drive & WD HD/Floppy Controller
Assembled and Tested $1069
Combo AT Parallel/Serial card $49

Complete 20 Meg AT with TTL Monitor $1699

Add $15 Shipping

33% IBM PC SPEEDUP

NEC V-20 5 MEG FOR 4.77 SPEED BOARDS ... $13.95
NEC V-20 8 MEG FOR TURBO BOARDS $19.95

AT Turbo Mainboard

- 6/8 Mhz Switchable
- O Wait State
- USA Writter BIOS
- K installed

$449

1.2 Meg AT Floppy Drive ..... $102.

MS—DOS 3.1

with GWBASICA $70

IBM Monitor Adapter

- Adpats your monitor
- AC cord to IBM Power Supply

6.75 ea. ..... 10 for $49.

Leading Edge Word Processor

For PC/XT or AT $28

Magnavox Professional

TTL Monitor $99

150 Watt Low Noise XT Powersupply ..... $69.

135 Watt Hard Disk Ready Powersupply ..... $65.

Arnold Company Also stocks a full line of RF & Computer Connectors. Call us for competitive pricing

TERMS: WE'VE MOVED TO SERVE YOU BETTER & FASTER!

Shipping $3.00 per order unless otherwise stated. C.O.D. add $1.90 extra. Texas residents add 6.25% tax. Certified Check or Money Order shipped same or next day. Personal checks may be held 10 days. C.O.D. orders welcomed. *Give us a call.* (214) 395-2922. We ship UPS ground. Air available at extra charge. Give us a call for fast friendly service. FREE flyer sent with each order. VISA/MC ADD 4%.

Non-defective returns subject to restock fee.

Copyright 1987 Arnold Co.

ARNOLD COMPANY · Computer/Electronic Supply · P.O. Box 512, Commerce, Texas 75428.

Math Pack #2

by Wynn Rostek

Welcome to the second Math Pack column. Last month I went over some of the pit-falls to watch for when using a computer for calculations. This month I'm going to present our first actual program, a program to solve systems of linear equations. This program that scientists and engineers need often, and also because it is used to finish the calculations started by many other programs that will be presented.

As I explained last time, I will present each program in *such a manner that you should* be able to code the program in your favorite language by reading the column and referring to the program presented, even if you don't speak "C."

The first order of business is to explain what a linear system of equations is. I'm sure you know what an equation is, it is a mathematical relationship between some variables or a variable and a constant. X = 3 is an equation even if it is a rather un-interesting one. The equation X = 3 describes a vertical line that passes through the point X = 3.

More common (remember algebra?) are lines that are neither vertical or horizontal, but have some slope. Lines like these have equations that look like 3Y + 4 = X. In this equation we have a relationship be*tween two variables and a con*stant. In general, an equation can show a relationship between any number of variables, not just 2.

If you remember conic sections from your algebra days, you may remember reading equations like "(X − 1) squared plus (Y − 3) squared equals 25." This equation is an example of a non-linear equation. A linear equation is one in which all the variables appear in the first power, you cannot have a variable squared, cubed or raised to any power greater than 1.

A linear equation cannot contain any of the transcendental functions either. No sin or cos, no log or square root, only variables raised to the first power such as X, Y, W etc.

Now knowing what a linear *equation is, we can discuss* what a system of linear equations is. In a system of linear equations, we have a set of equations that all contain a common set of variables. Each equation may contain some or all of the variables. The equations express different relationships between the variables. At this point, a simple example is due:

Jason, Bill, and Harry are friends. Twice Bill's age plus 3 times Jason's age minus Harry's age is 68. Jason's age plus Bill's age plus twice Harry's age is 91. Twice Bill's age plus Jason's age minus Harry's age minus 10 is 36. How old are Bill, Harry and Jason?

Remember these little torture devices from school? If you never caught on to the secret of linear systems, these kind of problems could cause you to pull out all your hair!

Let's set up the question in a form that is more easily worked with. Let Jason's age be represented by the variable J. Let Bill's age be represented by the variable B, and we'll use H for Harry's age.

With our representation selected, we can re-write the information.

2B + 3J − H = 68
J + B + 2H = 91
2B + J − H − 10 = 36

This is a system of linear equations! We now arrange them in a standard format. A variable must appear in the same column in each equation, and the constant must appear on the right side of the equals *sign. (I will be talking about* rows and columns a lot, to remember which way columns run think of the columns of a building, they run up and down.)

continued on page 262

Hacking The ST

continued from page 259

is positive. Sign extension is all that is needed to convert a 16 bit number to a 32 bit number. (2's complement numbering is a topic for another day.)

Moving a word into a data register does not result in sign extension, in this case the upper 16 bits of the register are left alone.

The next addressing mode we'll discuss is address register indirect. In address register indirect addressing, the specified register contains the address.

move.w (a0),d0

This instruction moves a word into register d0. The 32 bit value contained in register a0 is taken as an address. The word at this address is moved into d0.

We have also seen address register indirect with post-increment. When using this, the first portion of the operation is the same as address register indirect, but after the contents of the specified address register is used as an address, the CPU adds a small value to the contents.

The value added depends on the size of the data. For .b extenders, one is added, for word operations, (.w extenders) 2 is added and for long operations, (.l extenders) 4 is added. The number added is enough to point to the next piece of data.

Very similar to the address register indirect with post-increment addressing mode is address register indirect with pre-decrement. The big difference with this mode is that instead of incrementing the contents of the specified address register after it is used, pre-decrement means that the contents of the specified address register is decremented before it is used.

A typical instruction using address register indirect with pre-decrement addressing is:

move.w -(a0),d0 * move a word into d0

As in post-increment, the value used to decrement the register contents is determined by the operation size. A .b operation gives a decrement size of 1, .w gives 2 and a .l operation results in a decrement by 4. The above instruction subtracts 2 from the contents of a0 and takes the new value as the address where the word is read from.

The post-increment and pre-decrement versions of address register indirect addressing are complementary. Post-increment is used to search upwards through memory (start at low memory and work your way up) while pre-decrement is useful for working your way down through memory.

Our next addressing mode is address register indirect with displacement. Just like in address register indirect addressing, you specify an address register and the contents of that register is treated as the address. The difference here is that you also supply a 16 bit number which is added to the contents of the specified address register to generate the complete address.

Since a register holds 32 bits and the displacement is 16 bits, the displacement is sign extended to 32 bits before being added to the contents of the specified register. The following statement uses this mode for the source.

new: move.w 14(a0),d0 * fetch a word to d0

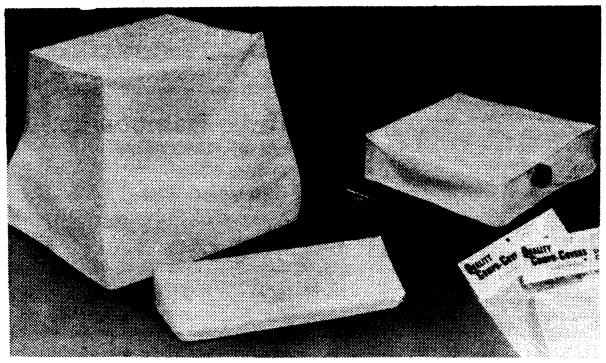
Now comes one of the most complex addressing modes, address register indirect with index. In this mode, the complete address is formed by adding together 3 separate quantities. The first part of the complete address is the contents of one of the address registers. The second part of the complete address is the contents of a second CPU register. The second register is known as the index register. You can use any of the address registers or data registers as the index register.

The third part of the complete address is an 8 bit number specified in the instruction. The 8 bit number is sign extended to 32 bits and added to the sum of the contents of the other two registers.

continued on page 262

DUST COVERS

ANTI-STATIC
HEAVY GAUGE BEIGE VINYL
100% MONEY BACK GUARANTEE



COMPUTERS		PRINTERS	
Apple IIE Mon, Sys, Duo Drives	15.95	Brother 15 9.95 W/T	10.95
Apple IIE Color Mon, Sys, Duo Dr.	15.95	Epson Rx or Fx-80/85	9.95
AT&T 6300 Mon, Sys & Keyboard	15.95	Epson Fx-100/185/286	12.95
Compaq Deskpro Mon, Sys & Kbd.	15.95	Gemini 10X or 15X (SG)	9.95
Leading Edge Model D	15.95	IBM Proprinter 9.95 XL	11.95
IBM PC or XT Mon, Sys & Kbd.	15.95	Imagewriter I 9.95 II	10.95
IBM AT Mon, Sys & Kbd.	15.95	Juki 6100 11.95 W/T	12.95
IBM PC Junior Mon, Sys & Kbd.	14.95	Okidata 82/92	9.95
Macintosh or Mac Plus	14.95	Okidata 83/84/93 W/T	11.95
Sanyo 550/555 Mon, Sys & Kbd.	14.95	Okidata 120/182/192	10.95
Tandy 1000/2000 Mon, Sys & Kbd.	15.95	Panasonic 1090/1091	10.95

QUALITY COMPU-COVERS ORDER FORM

Please Specify Exact Model: _____

_____

$2.00 Shipping Fee (N.Y. Residents Add Sales Tax) $ 2.00

Total Amount of This Order ..... $ _____

Name _____

Address _____

City _____ State _____ Zip _____

Over 300 different Covers for Computers, Printers, etc.
Call or Write for Complete Price List.

Quality Compu-Covers Inc.

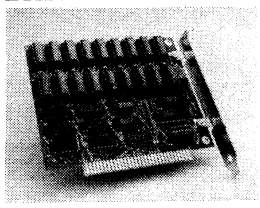
DEALERS INVITED

$2.00 Off Order with Ad

P.O. Box 649 • New City, N.Y. 10956 • (914) 634-9071

4-87

BREAK THE 640KB DOS BARRIER - $199.95



- ✔ More Room (up to 64KB more) to Run Application Software
- ✔ Installs RAM resident utilities above the operating system
- ✔ Multiple RAMdisks & Spoolers install above DOS

HiCard™ Memory Board

The HiCard™ 256/512KB Memory Board expands PCs and compatibles to a maximum of 896KB. RAMdisks and print spoolers are installed as resident utilities above the 704KB DOS Memory. RAM resident programs, like network drivers and DOS utilities also install in upper memory, freeing DOS to run your application software.

The HiCard™ is the only memory product to maximize **total** system capacity without modifying existing software.

Manufactured in the USA by

RYBS Electronics, Inc.

POB 4521, Boulder CO 80306
256KB $199.95
512KB $249.95
MC/VISA accepted

To Order: 1-800-825-8333 (tone, then) 444-4000.
Technical information, other inquiries 1-303-444-6073
DEALER INQUIRIES INVITED

Hacking The ST
continued from page 260

There is one item that should be mentioned, the index register can be of word size or long. When it is a word quantity, the word is sign extended to 32 bits. An instruction using this mode is shown below.

```
new: move.l 10(a0,d3.w),d0 *
     load a long to d0
```

The first operand uses this addressing mode. Let's look at the pieces that make up the complete address. The first item we encounter 10, is the 8 bit displacement. This is converted into a 32 bit 10 by adding 24 leading 0 bits to it. The 10 is added to the contents of register a0, which is the second part of the complete address. The d3.w is the third part of the address. To generate the third part of the address, the lower 16 bits of register d3 is sign extended to 32 bits and added to the partial address.

We now have added 3 32 bit numbers together to come up with the complete address. The last thing done is to get a long value (4 bytes) from memory at the address we just computed. The long value is then placed in register d0.

The next mode is absolute short addressing. In absolute short addressing, you supply a 16 bit number, which is sign extended to 32 bits to get the actual address. An instruction using this mode is:

```
top: move.w 2734,d0
```

This moves a word from memory location 2734 into register d0.

As you may have guessed, there is an absolute long addressing mode to go with the absolute short addressing mode. In absolute long addressing, you supply a 32 bit number, which needs no sign extension. An instruction to load d0 with a word from location 524367 is:

```
next: move.w 524367,d0
```

The next addressing mode we'll cover is program counter with displacement. This mode is very useful for branching. In this mode, you supply a 16 bit number which is added to the contents of the program counter to generate the address. An instruction using this looks like:

```
botm: bra -294(pc)
```

This instruction causes the program to jump back 294 bytes to get the next instruction. While this is a very useful mode, you rarely see it in this form. Because you normally use symbolic names instead of counting how many bytes back you need to jump, you normally write:

```
botm: bra top
```

Since the assembler knows where top is, it does the counting back for you and computes the value needed, then it translates top to −294(pc) for you.

Our next mode is program counter with index. This works the same way address register indirect with index does except that we use pc instead of an address register. This mode is shown here:

```
mung: beq 26(pc,d3.1)
```

I have seen very few systems where the assembler (or compiler) will generate this code, but the mode is there for those cases where it is needed.

The next mode is one that is very handy, and often used. In immediate mode addressing, the data is part of the instruction! The instruction shown below loads 27 into register d0.

```
esc: move.w #27,d0
```

The "#" character specifies the immediate mode. The CPU gets an instruction that says the source is the next 2 bytes (or 4 bytes).

This brings us to the last addressing mode, status register addressing. If you remember our discussion of flags being set by the compare instruction, you may have wondered where these flags were kept. The CPU lumps these flags together with some other indicators into something called a status register. Even though it is referred to as a status register, it is really only a collection bits.

There are a few operations such as logical AND and logical OR that can be performed on these bits. By using these instructions you can force some of these flags to known states.

The lower 8 bits of the status register is often called the ccr and it contains all of the user flags. The 68000 has a mode called supervisor mode. Some of the instructions of the 68000 can only be executed when the CPU is in the supervisor mode. The upper bits in the status register are associated with the supervisor mode while the lower 8 bits are associated with the user mode.

An instruction to set some of the user flags is:

```
all: ori.b #4,ccr
```

Motorola lumps some of the modes together into implied register mode. There is also a "quick" mode which is like immediate mode except that only a 3 bit positive number is allowed. If you are serious about 68000 assembly language programming, you may want to try and obtain a small card stock folder on the 68000. The only information I have about the folder is the number MC68000/AC REV 2 which appears in the upper right corner of the front cover.

The folder I have was included with the MC68000 16-bit microprocessor manual I received from Motorola. You may want to write to Motorola at: Motorola Semiconductors, 3501 Ed Bluestein Blvd. Austin, Texas 78721.

The folder and booklet supplied are very helpful for people struggling to understand the 68000, and are the final word on the operation of the 68000. ●

Math Pack #2
continued from page 260

After I arrange the equations to fit the rules above we have:

$$2B - 1H + 3J = 68$$
$$1B + 2H + 1J = 91$$
$$2B - 1H + 1J = 46$$

I now come to the juicy part. We are going to perform a series of operations known as "Gaussian Elimination" that leaves the system of linear equations in a form that makes solving for all the variables very simple.

Before we start in on the actual Gaussian Elimination, I need to discuss the kinds of operations that can be performed on a system of linear equations. You can exchange any two columns of the linear system, and you still have a linear system. As an example, assume I wanted to have Jason in the first column and Bill in the second column.

$$3J + 2B - 1H = 68$$
$$1J + 1B + 2H = 91$$
$$1J + 2B - 1H = 46$$

This is an equivalent linear system. If we solved for the three ages of the first set of equations and the second set of equations, we find the same answer for both.

We can also exchange any two rows and still have an equivalent linear system.

$$1J + 1B + 2H = 91$$
$$3J + 2B - 1H = 68$$
$$1J + 2B - 1H = 46$$

Another operation we can perform is scaling. We can multiply any equation by any number and still get the same answer when we are done.

$$3J + 2B - 1H = 68$$
$$2J + 2B + 4H = 182$$
$$1J + 2B - 1H = 46$$

(The above equation has been multiplied by 2.)

The last operation presented is the least intuitive of the bunch, but it is the critical operation in Gaussian Elimination. You can replace any equation (row) with the sum of that row and any other row in the system. As an example:

Replace the second row with the sum of the first and second row.

$$3J + 2B - 1H = 68$$
$$+ \quad 1J + 1B + 2H = 91$$
$$4J + 3B + 1H = 159$$

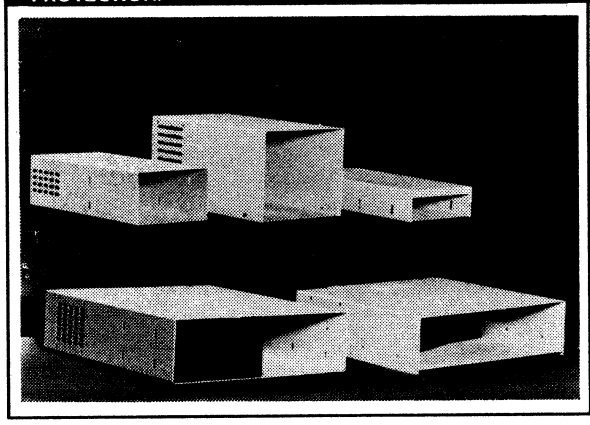
continued on page 269

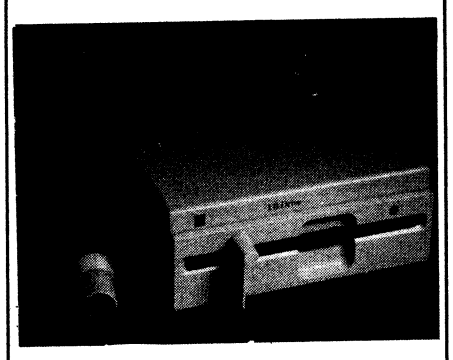
DISK DRIVE CASES		w/Power	Case Only
5¼" SINGLE	½ HT CASE	$ 49.95	$19.95
5¼" DUAL	½ HT CASE	$ 59.95	$21.95
5¼" HARD DISK	½ HT CASE	$ 89.95	
5¼" DUAL	F HT CASE	$ 69.95	$31.95
5¼" SINGLE	F HT CASE	$ 49.95	$21.95
5¼" HARD DISK	F HT CASE	$119.95	
3½" DUAL OR SINGLE CASE		$ 69.95	$26.95

CHIP RESISTANT FINISH. CASES ACCOMMODATE ALL STANDARD DRIVES. OVERCURRENT AND OVERVOLTAGE PROTECTION.

ATARI ST IB DRIVE

READS & WRITES PC 5¼" FORMAT





DEALERS WANTED

TERMS: Personal checks allow 14 days, COD, MO., Certified Checks . . . Credit Cards and 3.5% Shipping and handling: $3.00 plus current UPS or Parcel Post rates.

24 HOURS BBS # 503-292-1321

IB COMPUTERS 503-297-8425

1519 S.W. Marlow, Portland, OR 97225

FCC APPROVED CLASS B PLUG-IN COMPATIBLE

ATARI ST

PROVIDES LINK TO PC COMPUTERS! EASY TRANSFER OF FILES AS DRIVE B: JUST INSERT 5¼" DISKETTE AND RUN. FORMATS TOO! $239.00 COMPLETE. CABLE ONLY $35.

ATARI 520ST users:

Expand **memory** to 1, 2.5 or 4 MB ($129.00 - 0K, $179.00, $595.00 and $1055.00) with easy to **plug in** internal kit. Uses 256K DRAM's for 1 MB upgrade and 1Mb DIP's to go to 2.5 and 4MB. 32 reconfigurable sockets and 2 jumper blocks give you all the memory expansion possibilities you will ever need!
OR... 1MB with 256K ZIP's only $169.00.
All mem. upgrades: **NO SOLDERING** required!, 68000 CPU still easily accessible for soon to come ATARI© blitter and other enhancements
Add $30.00 for real time clock module!
20 MB **hard disk,** full SCSI implementation and many more features - 595.00
1 Mb DRAMs, 100ns DIPs - 31.00
For **more information** write or phone to:

tech-specialities Co.
909 Hodgkins Suite A
Houston, TX 77032 (713) 590-3738

1 year ltd. warranty - s/h: $5.00, HD $20.-
COD $3.00 - Texas residents + 7.125% -

MacroManager

Reviewed by Randy Holcomb

MacroManager is a desk accessory for the Atari ST series that is a welcome package in its usefulness and ease of use.

Like most desk accessories, MacroManager resides in memory and is available at the desktop by clicking the "MacroManager" entry. MacroManager provides several utilities in one: MacroAnalyst, a powerful calculator that has many of the features of a good hand-held pocket calculator (including business and scientific functions,) a card file/event log; an alarm clock/calendar; a note pad facility, a weekly planner and time accounting system, and a quick-and-dirty "typewriter" mode with some useful features. All of these functions are found in MacroManager.

Installing MacroManager is very easy to do and straightforward, with the documentation explaining the installation process in detail for both monochrome and color systems. Once installed on your boot disk, you must re-boot MacroManager in order to activate it on your ST.

The first time MacroManager is invoked a dialog box as shown in Figure 1 appears. This box allows you to reset the system date and time if necessary, and to determine the kind of modem being used with your system for the auto dialer feature which is available in MacroManager's card file/event log section. Once this dialog box has been satisfied, you are now presented with your major dialog box that MacroManager uses (Figure 2).

The calculator is quite interesting in the fact that it not only provides normal scientific calculator functions but has the ability to create an adding machine tape output by clicking the Tape print icon, and also provides for business calculation for loan payments, and performs operations in both algebraic and RPN (a la HP) modes.

The real heart of MacroManager is the electronic card file, where names, addresses, phone numbers (used by the auto dialer) are handled. When an entry is added to the card base, its date and time stamp are updated, which I find quite useful in determining the age of entries in the card file for possible purging and updating. It should be noted that, like most other desk accessories, MacroManager loads the data files required at boot time during accessory initialization, and depending on the number of entries in the various parts of MacroManager you may find that it starts taking up a good chunk of available memory if you are running on a 520ST. (The accessory itself takes up 70K; compiling a time report adds an additional 5K) but unless you have several hundred or thousand entries in the card database you shouldn't run into space problems. Another nice feature of MacroManager is the ability to print the contents of the card database with the whole card information, the first line and phone number, or the name and the next 3 lines in the database.

The Event Log behaves in much the same manner as the card file does and stores it in a separate database.

The Alarm Facility allows you to create entries in the Alarm database and interfaces with the weekly time planner. As long as your ST is powered

continued on page 272

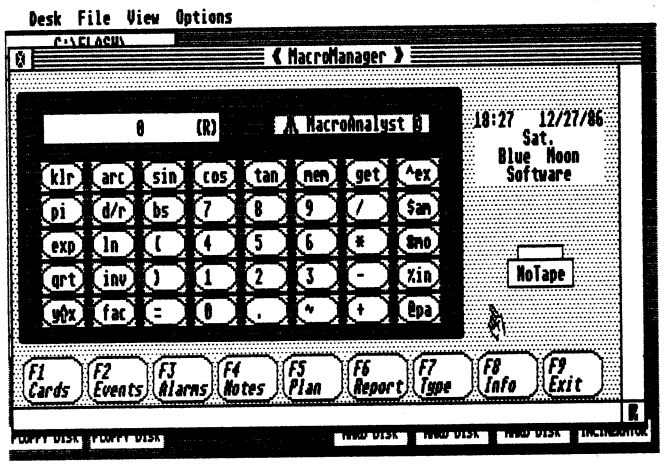


Figure 1. MacroManager Main Dialog Window

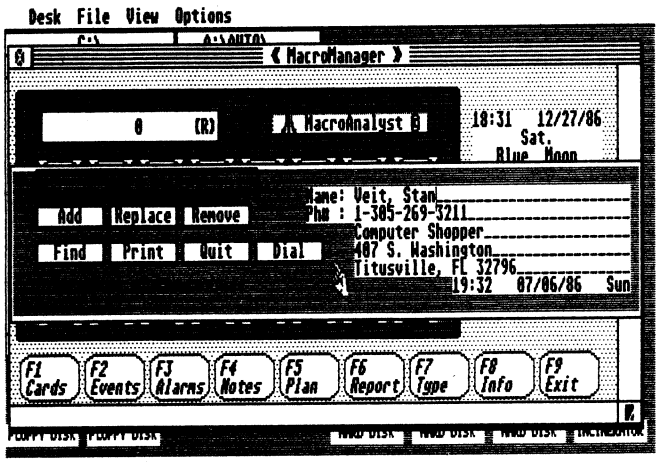
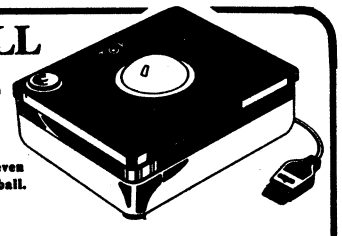


Figure 2. Card File Dialog Box

WICO TRACKBALL CONTROLLERS ONLY $19.95



Order by phone or mail. Specify which of the seven versions you want. Include $3.00 S&H per trackball. VISA/MC accepted. NY residents add sales tax.

ATARI—joystick—compatible version works on just about any computer joystick port that's ATARI—compatible, including:
COMMODORE 64
VIC-20 AMIGA
ALL ATARI 8-BIT
ATARI ST
TIMEX TS2068
1

2 ATARI ST - MOUSE
3 AMIGA - MOUSE
4 Radio Shack COCO
5 TI 99/4A
6 APPLE II, IIe, IIc
7 Odyssey Video Game

These incredibly low trackball prices are made possible because WICO sold off much of their consumer division products at liquidation prices.

All product is new. Some versions, like those for the Amiga and ST mouse ports, are made by Zebra modifying previous WICO versions. All products come with Zebra's 15—day money back guarantee and 1 year limited warranty.

The ST and AMIGA mouse versions, and AppleII versions have two buttons. All other versions have one.

All versions are fully compatible with software designed for their respective port function except the Apple II which includes a plug—in bus card and has little software support.

WICO is the world's largest designer and manufacturer of control devices for the commercial arcade games. If you've ever played an arcade video game, chances are you've experienced the superior control The pinpoint firing accuracy. The exceptional durability.

Quantities are limited to those on hand, so order now! These are special introductory prices. Prices are subject to change without notice.

ZEBRA SYSTEMS, INC.
78-06 JAMAICA AVE., WOODHAVEN, NY 11421, (718) 296-2385

Three new products for your Atari ST!

GFA BASIC
Which high level language for the Atari ST is as easy to program as BASIC yet has full access to system features and speed that rivals assembly language? The answer, *GFA BASIC!*

GFA BASIC

GFA BASIC resembles Modula-2 or Pascal code with the ability to accept parameters from the main program as well as using local variables. Also, various commands let you handle GEM drop-down menus. *GFA BASIC* stretches the definition of the BASIC computer language.

Color/Monochrome $79.95

GFA Compiler
GFA BASIC sets the standards for speed and simplicity, and the *GFA Compiler* adds even more power by converting your finished programs into compact, fast-running machine language files.

GFA Compiler

This 2-pass compiler is totally compatible with *GFA BASIC*. Through the most advanced techniques, your program is compiled in mere seconds!

GFA Compiler utilizes all the GEM features: windows, drop-down menus, and mouse control. The self-contained files composed by the *Compiler* operate without the aid of a runtime module, and you pay no royalties when selling a program compiled with the *GFA Compiler*.

Requires *GFA BASIC Interpreter.*
Color/Monochrome $79.95

TRIMbase
This Relational Data Management system is easy to use; no programming knowledge is required, yet it has the power to handle large amounts of complex data. As your application grows, *TRIMbase* continues to supply the efficient support you need.

TRIMbase

TRIMbase is capable of customizing your reports, relational operations (such as joining, subtracting, merging files, and partitioning files).

Color/Monochrome $99.95

MichTron
576 S. Telegraph, Pontiac, MI 48053
Orders and Information (313) 334-5700

GFA BASIC
continued from page 259

really easy, using the GFA BASIC command ALERT.

The procedure Intro simply provides a text block with an explanation of what GFA BASIC is and what this program attempts to do.

Procedure Default displays the four default windows available to GFA BASIC. They are opened, updated, and closed with clicks of the mouse, demonstrating the commands TITLEW, INFOW, OPENW, CLEARW, and CLOSEW. I should note that the documentation clearly states that the CLOSEW command must be used when a window display is completed, and that quick-clicking between four open windows can occasionally result in a crash. I tried coding it in about six different ways, but could not bulletproof the code. In actual practice, your programs will probably not call on four windows to open, update, and close in such quick succession.

Procedure Alert simply shows how alert boxes can be nested and branched, with minimal effort.

Procedure Gem demonstrates what is involved in creating an honest to goodness, movable, sizable window. It is difficult, to say the least, and probably beyond the scope of the beginner. But it can be done.

Procedure Cursor shows how the cursor can be cycled through eight ROM defaults. The command DEFMOUSE also supports user-defined mouse shapes, though this is not demonstrated here.

Procedure Position uses the command MOUSE to read the position of the mouse and print its coordinates.

Procedure Buttons reads the mouse buttons and prints their condition, using the command MOUSE.

Procedure Fonts demonstrates the command DEFTEXT, and shows the fonts, styles, and combinations available with it.

Procedure Graphmode shows how simply graphics can be handled with the GRAF MODE command. It puts small circle on the screen, controlled by the mouse. When you hold the left button down, you can draw with the circle shape.

Procedure Sprite demonstrates the SPRITE command, bouncing an animated block on the screen.

And that's all I wrote.

I had initially written this program on a monochrome ST system, and found upon running it in color that it did require modification—some fields were suddenly way out of range, text was printing in the wrong size, and the like. But the modifications were simple. The version of the program appearing here will run on all ST systems, mono or color. One advantage of the monochrome system while running GFA BASIC, by the way, is a 48-line text mode, wherein a nice big chunk of the program can be seen all at once. You can easily toggle between this mode and regular (23-line) mode.

If you own an Atari ST and you think BASIC has gotten a bum rap, you owe it to yourself to have a look at GFA BASIC. It just might be good enough to clear BASIC's name, at least as far as the Atari ST is concerned.

Remember, this will only happen if you try hard not to use GOTO—unless absolutely, positively necessary.

For more information contact Michtron, 576 South Telegraph, Pontiac, MI 48053. (313) 334-5700. ●

Listing 1

```
'
'-------------------------------------------------------------------------
'
'
' GFA BASIC DEMONSTRATION -- V 1.0
'
' by John J. Anderson, Computer Shopper Magazine
' CompuServe 76703,654
' Delphi      JOHNANDERSON
' Genie       J.J.ANDERSON
'
' Absolutely no rights reserved. Feel free to improve, distribute, ignore.
'
'
'---------------------------------------------------- create menu strip
'
Dim Bar$(90)    !   dimension string to hold menu text
For I=0 To 90   !   set up a loop
 Read Bar$(I)   !   read data from data field
 Exit If Bar$(I)="*****"   !until the marker is reached
Next I
Bar$(I)=""   !   tail blanks into string
Bar$(I+1)=""   !   ditto                    --actual menu text follows
Data  Desk , About this program
Data ----------
Data 1,2,3,4,5,6,""
Data  Intro , Introduction ,----------, quit ,""
Data  Windows , Default Windows , Alert Boxes , GEM Windows ,""
Data  Mouse , Cursor Definition , Mouse Position , Mouse Buttons ,""
Data  Graphics , Fonts , Graphmode , Sprites ,""
Data
Data ***
Menu Bar$()   !   activate menu
'
'---------------------------------------------------- main program loop
'
Do
 On Menu
 Gosub Menu   !   All the loop does is say "look at the menu and branch"
Loop
'
'---------------------------------------------------- reading menu
'
Procedure Menu   !   This procedure parses the menu selections
 If Bar$(Menu(0))=" About this program"   !   If this is the selection,
  Gosub About                            !   go to the corresponding procedure
 Endif
 If Bar$(Menu(0))=" Introduction "        !   ditto
  Gosub Intro
 Endif
 If Bar$(Menu(0))=" quit "
  System                                 !   the escape hatch
 Endif
 If Bar$(Menu(0))=" Default Windows "
  Gosub Default
 Endif
 If Bar$(Menu(0))=" Alert Boxes "
  Gosub Alert
 Endif
 If Bar$(Menu(0))=" GEM Windows "
  Gosub Gem
 Endif
 If Bar$(Menu(0))=" Cursor Definition "
  Gosub Cursor
 Endif
 If Bar$(Menu(0))=" Mouse Position "
  Gosub Position
 Endif
 If Bar$(Menu(0))=" Mouse Buttons "
  Gosub Buttons
 Endif
 If Bar$(Menu(0))=" Fonts "
  Gosub Fonts
 Endif
 If Bar$(Menu(0))=" Graphmode "
  Gosub Graphmode
 Endif
 If Bar$(Menu(0))=" Sprites "
  Gosub Sprite
 Endif                                   !   sure it's long, but it's easy, too
Return
'
'---------------------------------------------------- menu components
'
Procedure About   !   information box for the program
 Menu Off.        !   turn off menu inverse shading
 M$="MGA Basic Tutorial|Version 1.0|John J. Anderson|Computer Shopper"
 Alert 1,M$,1,"OKAY",B
Return
'
'---------------------------------------------------- introduction
'
Procedure Intro   !   a "pseudo-window", made out of a filled rectangle
 Menu Off
 Cls              !   clear the screen
 Deffill 1,0,0    ! define the fill
 Pbox 20,20,550,170   !   draw the box
 Deftext 1,4,0,6      !   define the text font          --text follows
 Text 30,30,"  GFA Basic is a structured Basic that brings the full power of"
 Text 30,40,"the Atari ST to the Basic programmer. It is easy to learn, fast,"
 Text 30,50,"features a powerful 'real-time' editor, and is packed with a"
 Text 30,60,"very complete command set for programming applications."
 Text 30,70," ---------------------------------------------------------- "
 Text 30,80,"  It comes with a run-only module that allows you to create "
 Text 30,90,"pseudo-stand-alone applications to share with other Atarians."
 Text 30,100,"Michtron and GFA Systemtechnik promise to release a true com-"
 Text 30,110,"piler soon which will allow for true stand-alone applications"
 Text 30,120,"and even greater speed. The package lists for $79.95."
 Text 30,130," ---------------------------------------------------------- "
 Text 30,140,"  This demo program shows just some of the power of GFA Basic."
 Text 30,150,"While using the program, you need never touch the keyboard."
 Text 30,160,"Just point and click, and have fun. - JJA"
 Menu Bar$()   !   reactivate menu bar
```

Listing 1 continued on page 267

Listing 1 GFA BASIC continued from page 266

```
Return
'
'------------------------------------------------------ default windows
'
Procedure Default    !   shows the four default windows
 Graphmode 1    !   make sure you've got the right graphics mode
 Menu Off
 Cls
 Titlew 1,"Window 1"    !   title for the window
 Infow 1," First default window"    !   subtitle for the window
 Titlew 2,"Window 2"
 Infow 2," Second default window"
 Titlew 3,"Window 3"
 Infow 3," Third default window"
 Titlew 4,"Window 4"
 Infow 4," Fourth default window"
 Openw 1
 Q=1    !   initialize variable that tracks active windows
 Hidem    !   do not display mouse
 Repeat    !   start a loop
  K=Mousek    !   read the mouse buttons
  Print At(2,2);"Left mouse button opens windows,"
  Print At(2,3);"right button closes them."
  Print At(2,4);"Press both buttons together"
  Print At(2,5);"to return to menu..."
  If K=1    !   see if left mouse button is being pressed
   Clearw Q    !   clear the contents of current window
   Inc Q    !   increment Q by 1, a neat GFA Basic command
   If Q>4    !   clip the counter within range
    Q=1
   Endif
   Openw Q    !   open new window q
  Endif
  If K=2    !   see if right mouse button is being pressed
   Inc Q
   If Q>4
    Q=1
   Endif
   Closew Q    !   close current window
  Endif
 Until K=3
 For I=1 To 4
  Closew I    !   close all windows
 Next I
 Closew 0    !   return to normal screen
 Menu Bar$()    !   reactivate menu bar
 Showm    !   reactivate mouse
Return
'
'------------------------------------------------------ alert boxes
'
Procedure Alert    !   Somewhat whimsical look at branching windows
 Menu Off
 B=0    !   initialize counters
 C=0
 Alert 2,"Are you impressed|with GFA Basic|so far?",1,"yes|no",B
 If B=2
  Alert 2,"Come on. You must|have seen something|you liked.",1,"yes|no",C
 Endif
 If B=1 Or C=1
  Alert 1,"Glad to hear it.",0,"okay",B    !   person is enthusiastic
 Endif                                     !   or at least non-hostile
 If C=2
  Alert 3,"Trash disk",0,"yes",B    !   person deserves a scare
  System                            !   don't worry, we're not
 Endif                              !   really going to trash anything
Return
'
'------------------------------------------------------ GEM windows
'
Procedure Gem    !   Shows you a sizable, movable window
 Menu Off    !   and the effort it requires
 Cls
 Attr=Windtab+2    !   set attributes for window
 Xpos=Windtab+4
 Ypos=Windtab+6
 Width=Windtab+8
 Height=Windtab+10
 Dpoke Attr,&HFFF    !   poke it in there
 Dpoke Xpos,20
 Dpoke Ypos,20
 Dpoke Width,350
 Dpoke Height,120
 Titlew 1,"GEM WINDOW"    !   name it
 Infow 1,""    !   no info box this time
 On Menu Message Gosub Message    !   read actions on window
 Openw 1    !   open it
 Clearw 1    !   clear it
 Repeat    !   fill it with randomly plotting lines
  Color Rnd+0.5
  Line Rnd*640,Rnd*400,Rnd*640,Rnd*400
  On Menu
 Until M=22    !   until user clicks on the close button
Return
Procedure Message    !   handles sizing, placement, and closure
 M=Menu(1)
 If M=22    !   closure
  Closew 1
  Closew 0
  Menu Bar$()
 Endif
 If M=23    !   full-size button has been pressed
  @Setxybh(0,10,630,180)
 Endif
 If M=27 Or M=28    !   user is custom-sizing window
  @Setxybh(Menu(5),Menu(6),Menu(7),Menu(8))
 Endif
Return
Procedure Setxybh(X,Y,B,H)    !   nasty little chunk of code
 Dpoke Gintin,Dpeek(Windtab)    !   to handle window size
 Dpoke Gintin+2,4    !   and placement
 Gemsys 104
 Xa=Dpeek(Gintout+2)
 Ya=Dpeek(Gintout+4)
 Ba=Dpeek(Gintout+6)
 Ha=Dpeek(Gintout+8)
 Get Xa,Ya,Min(Ba,B),Min(Ha,H),Temp$
 Closew 1
 Dpoke Xpos,X
 Dpoke Ypos,Y
 Dpoke Width,B
 Dpoke Height,H
 Openw 1
 Clearw 1
 Dpoke Gintin,Dpeek(Windtab)
 Dpoke Gintin+2,4
 Gemsys 104
 Put Dpeek(Gintout+2),Dpeek(Gintout+4),Temp$
 Clearw 1
 Closew 0
 Openw 1
Return    !   wish it were easier than this, folks...
'
'------------------------------------------------------ changing cursor
'
Procedure Cursor    !   lets you click through ROM cursor shapes
 Menu Off
 Cls
 Deftext 1,0,0,6
 Text 10,160,"Defmouse { }"
 Text 10,170,"Left mouse button changes cursor,"
 Text 10,180,"right mouse button for menu..."
 Repeat    !   set up a loop
  Pause 10    !   put a brake in to damp selection, as DEFMOUSE works
  Mouse X,Y,K    !   so fast a single click often jumps selections
  If K=1    !   read mouse, see if left button is pressed
   Inc Q    !   if it is, increment counter by 1
   If Q>7    !   clip counter into range
    Q=0
   Endif
   Text 90,160,Q    !   print out the value
   Defmouse Q    !   define the mouse as that value
  Endif
 Until K=2    !   until the right mouse button is finally pressed
 Menu Bar$()
Return
'
'------------------------------------------------------ reading mouse
'
```

Listing 1 continued on page 268

Listing 1 GFA BASIC continued from page 267

```
Procedure Position    !   a real simple one, reading out mouse X and Y
  Menu Off            !   coordinates
  Cls
  Deftext 1,0,0,6
  Text 10,170,"Mouse {  ,  ,0}"
  Text 10,180,"Right mouse button for menu..."
  Repeat
    Mouse X,Y,K
    Text 67,170,X    !   print out the actual value
    Text 99,170,Y    !   ditto
  Until K=2          !   these are sometimes flaky and I don't know why
  Menu Bar$()
Return
'
'------------------------------------------------------- reading buttons
'
Procedure Buttons    !   another simple one to show mouse button presses
  Menu Off
  Cls
  Hidem    !   hide the mouse
  Deftext 1,0,0,6
  Text 10,170,"You are pressing mouse button {  }"
  Text 10,180,"Press both buttons together to return to menu..."
  Repeat
    Mouse X,Y,K       !   read the mouse buttons
    Text 263,170,K    !   print out the actual value
  Until K=3           !   until both buttons are pressed
  Showm               !   reactivate the mouse
  Menu Bar$()         !   reactivate the menu bar
Return
'
'------------------------------------------------------- changing fonts
'
Procedure Fonts    !   lets the user click through available fonts
  Menu Off
  Cls
  Deftext 1,0,0,6
  Text 10,160,"Left mouse button clicks through styles,"
  Text 10,170,"right mouse button clicks through sizes."
  Text 10,180,"Press both buttons together for menu."
  S=0    !   initialize counters
  G=4
  Hidem    !   hide the mouse once again, or it flickers
  Repeat    !   set up a loop
    Pause 10    !   once again, a brake to damp the action
    Mouse X,Y,K
    Text 10,70,T$
    If K=1
      Inc S    !   increment counter on clicks of left mouse button
      Deffill 0,1,1    !   set up a filled rectangle to erase
      Pbox 8,30,350,80    !   previous entries
    Endif
    If S=0    !   this section works same way as menu parser
      T$="normal text"    !   hopefully self-explanatory
    Endif
    If S=1
      T$="boldface"
    Endif
    If S=2
      T$="light"
    Endif
    If S=3    !   clip the counter to next meaningful value
      S=4
      T$="italic"
    Endif
    If S=5
      T$="boldface italic"
    Endif
    If S=6    !   clip the counter
      S=8
      T$="underlined"
    Endif
    If S=9    !   clip the counter once again
      S=16
      T$="bordered"
    Endif
    If S=17
      T$="boldface bordered"
    Endif
    If S=18    !   clip
      S=20
      T$="bordered italic"    !   looks weird, but what the heck
    Endif
    If S=21    !   clip back to the beginning
      S=0
      T$="normal text"
    Endif
    If K=2
      Inc G
      Deffill 0,1,1    !   set up blanking rectangle once again
      Pbox 8,30,350,80
    Endif
    If G=5    !   now we're doing the same thing again, but this
      G=6     !   time we're looking at the right mouse button
    Endif     !   and incrementing a different counter
    If G=7
      G=13    !   calls for a lot of clipping, doesn't it?
    Endif
    If G=14    !   we're changing font sizes here
      G=32
    Endif
    If G=33
      G=4
    Endif
    Deftext 1,S,0,G    !   this is the line that actually does it
  Until K=3
  Menu Bar$()
Return
'
'------------------------------------------------------- graphmode demo
'
Procedure Graphmode    !   draws and "undraws" a circle with the mouse
Menu Off
Cls
Deftext 1,0,0,6
Text 10,160,"Graphmode {mode}"
Text 10,170,"1=replace 2=transparent 3=xor 4=reverse transparent"
Text 10,180,"Left mouse button to draw, right mouse button for menu..."
Repeat    !   set up a loop
  Mouse X,Y,K
  If K=1    !   if the left button is pressed, circle "draws"
    Graphmode 2
  Else
    Graphmode 3    !   otherwise it just sits there, undrawing itself
  Endif
  Circle X,Y,50    !   this line actually plots it
  Circle X,Y,50    !   unplots it unless button is pressed
Until K=2
Graphmode 1    !   reset graphics mode
Menu Bar$()
Return
'
'------------------------------------------------------- sprite demo
'
Procedure Sprite    '   demonstrates a bouncing sprite
  Menu Off
  Cls
  Deftext 1,0,0,6
  Text 10,190,"Hold right mouse button down for menu..."
  A$=Mki$(1)+Mki$(1)+Mki$(0)    !   packs string with sprite data
  A$=A$+Mki$(0)+Mki$(1)
  For I=1 To 16
    A$=A$+Mki$(0)+Mki$(65535)  !   65535 is a solid line
  Next I                       !   read it in 16 times
  Hidem                        !   and it becomes a block
  Repeat
    Mouse X,Y,K
    For I=0 To 190    !   throw it down
      Sprite A$,190,I
    Next I
    For I=190 To 0 Step -1    !   bounce it up
      Sprite A$,190,I
    Next I
  Until K=2    !   until the right mouse button is pressed
  Menu Bar$()
  Showm
Return
'
'
'------------------------------------------------------- END
```

Math Pack #2
continued from page 262

$$3J + 2B - 1H = 68$$
$$4J + 3B + 1H = 159$$
$$1J + 2B - 1H = 46$$

When we are trying to solve a linear system (a system of linear equations) we can perform any sequence of the above operations and we still get the same answer!

We will be trying to solve 3 equations with 3 unknowns. (B, J and H are our 3 unknowns, and the three equations are the three rows of our linear system)

When we perform Gaussian Elimination on the linear system, it is customary to drop the variables and only worry about the coefficients. After we do this, our system looks like:

$$2 - 1 + 3 = 68$$
$$1 + 2 + 1 = 91$$
$$2 - 1 + 1 = 46$$

What we are trying to find is a simple equation that looks like B = 15, or whatever B really equals. The operations we can use are exchange two rows or two columns, scale a row, or replace a row with the sum of that row and another row.

The numbers on the main diagonal (2, 2, and 1 in our example) will be referred to as the pivot elements. In order to get a simple equation like H = 19, we have to eliminate J and B from the equation. The third equation in our linear system needs to look like OB + OJ + XJ = some number.

To arrive at this happy state of affairs, we will "zero out" all elements in a column below the pivot element. When this is completed, the linear system will be "upper triangular." An upper triangular system has all 0s below the main diagonal.

The time has come for an example to show how this is done. We will start by "zeroing out" column 1. The elements in the first column have to be "zeroed out" one row at a time. I am going to "zero out" the 1B from row 2. (Remember that the B is not written so that we can concentrate on the actual numbers.)

The steps in "zeroing out" an element are simple, and are:

1. Scale the row containing the element being eliminated.
2. Replace the row with the sum of the row and the pivot row.

When you go to scale the row, we don't use just any scale factor, we pick a special scale factor that allows us to eliminate the element when we perform step 2.

What we are going to do is replace 1 + 2 + 1 = 91 by scaling this equation and adding the scaled equation to 2 - 1 + 3 = 68. The result of the addition should have a 0 as the first element. Since I am adding the scaled equation to 2 - 1 + 3 = 68, the scaled equation must start with -2.

Since I need a better way to talk about the numbers in the linear system, I will introduce a convention here that allows us to talk about a number without worrying about what the actual number is.

When we store the data that makes up the linear system in our computer, we will store it

continued on page 272

Listing 1

```
/* 01/01/87 Started work on this file */

#include <stdio.h>
#include <math.h>

/* this file contains the routines that allow us to solve systems */
/* of linear equations                                            */

/* this program expects to be passed the name of a file that contains */
/* the size of the array and all the entries in the array             */
/* by keeping this information on disk, and leaving the answer on the */
/* disk, we can chain the programs together with batch files          */

/* the data in the input should consist of a series of numbers   */
/* one per line.  The first number (an integer) is the number of */
/* rows in the augmented matrix.  The second integer is the      */
/* number of columns in the matrix. The entries that follow are  */
/* the actual data.  You should have the first number from the   */
/* first row, then the second number from the first row, then    */
/* the third number from the first row, etc.  You then have the  */
/* first number from the second row, second number from the      */
/* second row and so forth                                       */


/*****************************************************************/
/*                                                               */
/* Notes for people adapting this program...                     */
/*                                                               */
/*  1. You will have to write and link in your own waitkey()     */
/*     This function prints hit any key to continue, then it     */
/*     waits for the operator to hit a key.                      */
/*                                                               */
/*  2. Watch all the printf statements that handle floating      */
/*     point values, the format specification varies from        */
/*     compiler to compiler.                                     */
/*                                                               */
/*  3. Watch for funtions that return VOID, some compilers want  */
/*     to see void, while other require #define VOID int        */
/*     Alcyon doesn't recognize void as valid, so I have added   */
/*     #define VOID int to my stdio.h file                       */
/*                                                               */
/*  4. math.h is only used because it contains the declaration   */
/*     double fabs(); which keeps you from having to declare it  */
/*     in all the functions where it is used...                  */
/*                                                               */
/*  5. watch the value you use for small, most systems only      */
/*     offer 4 or 5 digits, use small /= 1000.0 for them         */
/*                                                               */
/*****************************************************************/

/* these defines control the size of the system we can solve   */
/*                                                             */
/* MAXCOL should be one larger than MAXROW                     */
/*                                                             */
/* it would be nice if all C preprocessors allowed:            */
/* #define MAXCOL (MAXROW + 1)                                 */
/* but many (most) do not.  Oh well...                         */
/*                                                             */

#define MAXROW 10              /* number of rows in the array    */
#define MAXCOL 11              /* number of columns in the array */

/* this define determines if we do diagnostics or not */
#define test TRUE

/* a few useful globals */
double a[MAXROW][MAXCOL];      /* our array of floating point numbers  */
double u[MAXROW][MAXCOL];      /* this holds our upper triangle matrix */
double result[MAXCOL];         /* the solution to the system           */
int arb[MAXCOL];               /* flags for arbitrary variables        */

/* here are some of the globals used for solving the linear system    */
int mxrow;                     /* last valid row number               */
int mxcol;                     /* last valid column number            */
int np;                        /* indicates limits for non-regular systems */
double small;                  /* we test for close to 0.0, not equal to 0.0 */
                               /* this is our close to 0.0 value      */
                               /* we need to do this because:         */
                               /* there is about a one in a million chance */
                               /* of a calculation yielding exactly 0.0    */
                               /* ain't discrete math wonderful?      */

/* these globals are general purpose */
FILE *dfile;                   /* diagnostic message file             */

main(argc,argv)

int argc;
char *argv[];

{
FILE *fopen();
FILE *infile;
FILE *outfile;
double ab;
int numrow;
int numcol;
int numchar;
int r;
int c;
double ftemp;
char outnme[80];
int i;
int sol;
int narb;

/* the first thing we need to do is see if we got a file name */
if (argc < 2)
    {
    printf("Sorry, you must supply a file name.\n");
    printf("Program is terminating.\n");
    waitkey();
    exit(-1);
    }

/* now we open the source file for reading */
infile = fopen(argv[1],"r");
if (infile == NULL)
    {
    printf("Unable to open %s for reading.\n",argv[1]);
    printf("Program is terminating.\n");
    waitkey();
    exit(-1);
    }

/* now that the file is open, read in the size of the array */
numchar = fscanf(infile,"%d",&numrow);
if (numchar < 1)
    {
    printf("Row size missing.\n");
    printf("Unable to continue, program is terminating.\n");
    waitkey();
    exit(-1);
    }
/* show the operator the data */
printf("%d rows\n",numrow);

numchar = fscanf(infile,"%d",&numcol);
if (numchar < 1)
    {
    printf("Column size missing.\n");
    printf("Unable to continue, program is terminating.\n");
    waitkey();
    exit(-1);
    }
/* show what we got */
printf("%d columns\n",numcol);

/* now we check to see if the size is valid */
mxrow = numrow - 1;
mxcol = numcol - 1;

/* first we see if it is too small */
if (mxrow < 1)
    {
    printf("%d rows specified.\n",numrow);
    printf("Invalid size for linear system.\n");
    printf("Program is terminating.\n");
    waitkey();
    exit(-1);
    }

if (mxcol < 2)
    {
    printf("%d columns specified.\n",numcol);
    printf("Invalid size for linear system.\n");
    printf("Program is terminating.\n");
    waitkey();
```

Listing continued on page 270

Math Pack Listing continued from page 269

```
    exit(-1);
    }

/* now we see if it is too large */

if (numrow > MAXROW)
    {
    printf("%d rows specified.\n",numrow);
    printf("This program works with %d rows maximum.\n",MAXROW);
    printf("Change MAXROW define and recompile.\n");
    printf("Program is terminating.\n");
    waitkey();
    exit(-1);
    }

if (numcol > MAXCOL)
    {
    printf("%d columns specified.\n",numcol);
    printf("This program works with %d columns maximum.\n",MAXCOL);
    printf("Change MAXCOL define and recompile.\n");
    printf("Program is terminating.\n");
    waitkey();
    exit(-1);
    }

/* now we open the diagnostic file if needed */
if (test)
    {
    dfile = fopen("TRACE.DAT","w");
    if (dfile == NULL)
        {
        printf("Unable to open TRACE.DAT for writing.\n");
        printf("Program is terminating.\n");
        waitkey();
        exit(-1);
        }
    }

/* since we made it here, we can start reading in the data */
/* we assume that the data is in row major format          */

/* the number below is supposed to be much larger than any */
/* number in your matrix, if your numbers are very large,  */
/* you may need to increase the value here                 */
small = 1000000.0;

r = 0;
FOREVER
    {
    c = 0;
    FOREVER
        {
        numchar = fscanf(infile,"%F",&ftemp);
        if (numchar < 1)
            {
            printf("Attempted to read array[%d][%d].\n",r,c);
            printf("Data not present.\n");
            printf("Unable to continue, program is terminating.\n");
            waitkey();
            exit(-1);
            }
        /* show the data */
        printf("%F\n",ftemp);
        /* we also keep track of the smallest value */
        ab = fabs(ftemp);
        if (ab < small) small = ab;
        a[r][c] = ftemp;
        c++;
        if (c > mxcol) break;
        }
    r++;
    if (r > mxrow) break;
    }

/* since there is NO chance of a floating point number being 0.0 ,  */
/* we test for it being close enough to 0.0 instead.                */
/* if you have floating point numbers with less than 64 bits, you   */
/* will need to use a number like 1000.0 in the next line.          */
/* you may also have to change this line if you are working with    */
/* really large or really small numbers in your matrix.             */
small /= 100000.0;

/* we also compute limits to handle non-regular systems */
if (mxcol > mxrow) np = mxcol;      /* normal system */
else np = mxrow - 1;

/* we can close the input file, we are done with it now */

fclose(infile);

/* now that we have all the data, do Gaussian elimination */
printf("Ready to start gaussian elimination.\n");
gaussian();

/* save a copy of the upper triangular matrix */
save();

/* now we do backward substitution */
printf("Ready to start backward substitution.\n");
sol = backward();

/* save a copy of the matrix now */
save();

/* now we save the results to a file */
printf("Saving results.\n");

/* we start off by creating the file */

/* first we decide which name to use */

if (argc > 2)
    {
    /* user supplied an output file name */
    strcpy(outnme,argv[2]);
    }
else
    {
    /* we use the default file name */
    strcpy(outnme,"LINSYS.SOL");
    }

/* now we open the file */
outfile = fopen(outnme,"w");
if (outfile == NULL)
    {
    printf("Unable to open %s for writing.\n",outnme);
    printf("Program is terminating.\n");
    waitkey();
    exit(-1);
    }

/* actually store the data */
for (i = 0;i < np;i++) fprintf(outfile,"%F\n",result[i]);

/* now we show the results */
if (!sol) fprintf(outfile,"System inconsistant\n");

/* see if we have any arbitrary variables */
narb = 0;
for (i = 0;i < MAXCOL;i++) if (arb[i]) narb++;
if (narb == 0) fprintf(outfile,"Solution unique\n");
else fprintf(outfile,"%d arbitrary variables.\n",narb);

/* if there are arbitrary variables, show which ones they are */
for (i = 0;i < MAXCOL;i++) if (arb[i]) fprintf(outfile,"%%d is
arbitrary\n",i);


/* for arbitrary variables, show the basis vectors */


/* and close the results file */
fclose(outfile);

/* and last but not least, close the diagnostics file */
if (test) fclose(dfile);
}

/* from here down we have the routines that support the solution */

VOID gaussian()

/* this routine performs gaussian elimination on the matrix */
{
int i;
double ab;

for (i = 0;i < np;i++)
    {
    docol(i);     /* partial pivoting */
    ab = fabs(a[i][i]);
    if (ab < small) a[i][i] = 0.0;
    else zrocol(i);
    }
}


VOID docol(col)

int col;

/* this routine handles a single column   */
/* we perform partial pivoting            */
{
double ab;
double big;
int large;
int i;

/* first we find the largest element */
big = fabs(a[col][col]);
large = -1;
for (i = col;i <= mxrow;i++)
    {
    ab = fabs(a[i][col]);
    if (ab > big)
        {
        big = ab;
        large = i;
        }
    }

/* now that we have found the largest value */
/* we can do the partial pivot if need be   */
if (large != -1) swaprow(col,large);
}


VOID swaprow(source,dest)

int source;
int dest;

/* this routine is used to swap two rows */
/* elements below source column are 0.0  */
{
double temp;
int col;

for (col = source;col <= mxcol;col++)
    {
    temp = a[source][col];
    a[source][col] = a[dest][col];
    a[dest][col] = temp;
    }
}



VOID zrocol(col)

int col;
```

Listing continued on page 271

Math Pack Listing continued from page 270

```
/* this routine is used to do the actual column reduction */
{
int row;
double fac;

for (row = col + 1;row <= mxrow;row++)
    {
    /* compute the scale factor */
    fac = -1.0 * (a[col][col]/a[row][col]);

    /* scale the row */
    sclrow(row,col,fac);

    /* and sum the rows */
    sumrow(row,col);
    }
}


VOID sclrow(row,col,fac)

int row;
int col;
double fac;

/* this routine is used to scale a row of the matrix */

/* elements left of col are already 0.0                */
{
int i;

for (i = col + 1;i <= mxcol;i++) a[row][i] *= fac;
}


VOID sumrow(row,col)

int row;
int col;

/* this routine is used to add two rows */
{
int i;

a[row][col] = 0.0;
for (i = col + 1;i <= mxcol;i++) a[row][i] += a[col][i];
}


int backward()

/* this routine is used to perform backward substitution    */
/* this routine returns FALSE if the system is inconsistant */
{
int i;
int numarb;
int hi;
int lo;
int solution;


/* show what we are starting with */
save();


/* the first thing we do is to take care of the obvious arbitrary */
/* variables, when we have more variables than equations          */
for (i = 0;i <= mxcol;i++) arb[i] = FALSE;

numarb = (mxcol - 1) - mxrow;
if (numarb > 0) for (i = (mxrow + 1);i < mxcol;i++) arb[i] = TRUE;

/* now we can start solving for the variables */
/* find the upper limit first */
if (mxrow < (mxcol - 1)) hi = mxrow;
else hi = mxcol - 1;

for (i = hi;i > -1;i--)

    {
    solution = subl(i);
    if (!solution)
        {
        /* looks like there is no solution, the system is inconsistant */
        printf("System is inconsistant.\n");
        printf("No solution space exists.\n");
        waitkey();
        return(FALSE);
        }
    }

/* it is still possible for the system to be inconsistant */
/* we need to check here for it...                        */
if (mxrow < mxcol)
    {
    /* under this circumstance we have a solution */
    printf("Solution found.\n");
    return(TRUE);
    }

for (i = mxcol;i <= mxrow;i++)
    if (result[i] != 0.0);
        {
        /* oops, inconsistant */
        printf("System is inconsistant.\n");
        printf("No solution space exists.\n");
        waitkey();
        return(FALSE);
        }
}
```

```
int subl(row)

int row;

/* this routine is used to solve for a single variable */
/* or column in our backward substitution              */
{
double ab;

/* first we have to test to see if the variable is arbitrary */
/* we check the pivot element for a value of 0.0             */
ab = fabs(a[row][row]);
if (ab < small)
    {
    /* looks like the pivot element is zero */
    ab = fabs(a[row][mxcol]);
    if (ab < small)
        {
        /* looks like the value is 0.0 also */
        arb[row] = TRUE;
        return(TRUE);
        }
    else return(FALSE);
    }

/* since we made it here, it's not an arbitrary variable */
/* and the system is not inconsistant (at least not yet) */
/* so we can now solve for the value of the variable     */
result[row] = a[row][mxcol]/a[row][row];
ab = fabs(result[row]);
if (ab < small)
    {
    /* we computed the variable to be equal to 0.0 */
    result[row] = 0.0;
    return(TRUE);
    }

/* since we made it here, the variable is not */
/* equal to 0.0 so we have to subtract the    */
/* variable from the other equations          */
if (row == 0) return(TRUE);

/* we have to factor the variable out of the other equations */
subvar(row,result[row]);
return(TRUE);
}


VOID subvar(row,value)

int row;
double value;

/* we have to factor out the variable of the other equations */
/* above this one in the matrix                              */
{
double diff;
double temp;
double ab;
int col;

col = row;
row--;        /* start above the row just solved */

FOREVER
    {
    diff = value * a[row][col];
    temp = a[row][mxcol] - diff;
    ab = fabs(temp);
    if (ab < small) temp = 0.0;
    a[row][mxcol] = temp;
    a[row][col] = 0.0;
    row--;
    if (row < 0) break;
    }
}


VOID show()

/* this routine is used to show the array              */
/* it is a handy diagnostic if you are having problems */
{
int r;
int c;

for (r = 0;r <= mxrow;r++)
    {
    for (c = 0;c <= mxcol;c++) printf("%-12F",a[r][c]);
    printf("\n");
    }
printf("\n\n");
waitkey();
}


VOID save()

/* this routine is used to save the array contents to disk */
/* it is a handy diagnostic if you are having problems     */
{
int r;
int c;

/* see if we are trouble shooting */
if (!test) return;

for (r = 0;r <= mxrow;r++)
    {
    for (c = 0;c <= mxcol;c++) fprintf(dfile,"%-12F",a[r][c]);
    fprintf(dfile,"\n");
    }
fprintf(dfile,"\n\n");
}
```

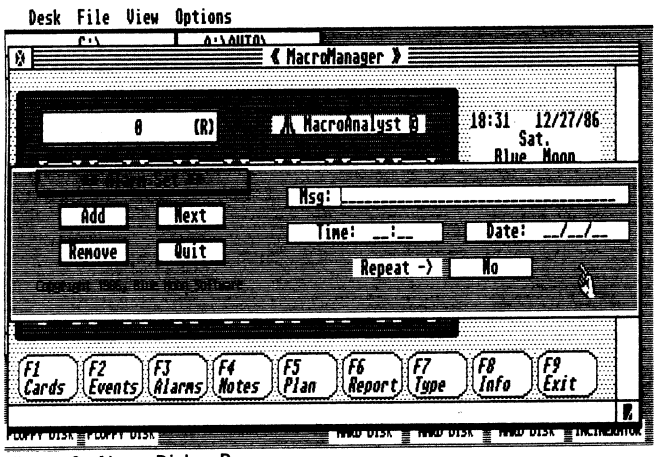


Figure 3. Alarm Dialog Box

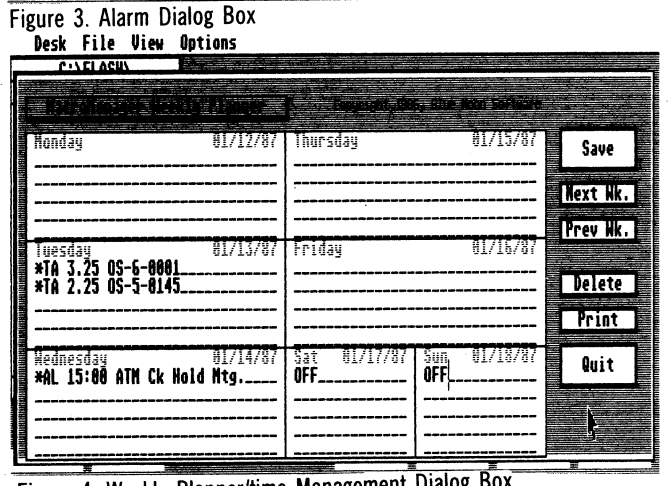


Figure 4. Weekly Planner/time Management Dialog Box.

WHEN
WHOLESALERS
WHOLESALE
THEY CALL US
WHY DON'T YOU?

DEALERS WELCOME
AMERICAN COMPUTER
RESOURCES, INC.
39 DAVENPORT STREET
STAMFORD, CT 06902
(203) 357-9024

* IBM * COMPAQ * AT&T *
* EPSON * SEAGATE * TOSHIBA *
* AMDEK * FOUNTAIN * IRWIN *
* OKIDATA * PGS * AST *
*HERCULES * SYSGEN *
*MANY MORE *
*HUGE INVENTORY *
*FAST DELIVERY *

WE BUY USED EQUIPMENT!
WE BUY EXCESS INVENTORIES!
WORD PROCESSORS * MINI COMPUTERS *
MICROS

MacroManager
continued from page 263

up MacroManager will throw up a dialog box when the alarm time and date have been reached. To shut the alarm off you have to click it off or hit RETURN. This is nice if your ST is on constantly and you use it constantly; otherwise its useability is limited as it won't queue up events that have passed by after you have turned on your ST! The alarm entry can be set for either one-shot or recurring processing.

The Note Pad is a quick and easy way to edit small ASCII files without having to fire up a word processor or text editor: just click or use function key for the note pad and you get this nice large dialog box into which you can enter your text, along with the pathname of the file to save the text in, as well as options to print the contents of the note pad and scroll the pages forward and back. Each note pad can hold just a shade over 4.5K fully loaded (60 lines of 70 characters) and you can have as many note pads as you wish (disk size being the only limiting factor).

The Weekly Planner is unique in that it has hooks to the alarm system and the time analysis section. What you do is set up a weekly itinerary and enter items on the weekly planner dialog box and proceed to enter the items in the appropriate day. If you prefix the entry with "*AL" MacroManager will send the entry to the alarm system and will force the alarm up when the date is reached and it won't go away until you acknowledge it. You can optionally attach a time for the alarm to appear and MacroManager will automatically set up the alarm to fire off at the proper date and time. In both cases, any text entered in the weekly planner will be shipped off to the alarm database and will be displayed accordingly.

For time management entries, you prefix the entry with a "*TA" followed by a time unit and the project code or name. Each project code can be up to 24 characters in length and the time accounting section can compile up to 200 unique project id's per report. (A word of caution MacroManager treats uppercase and lowercase characters in time accounting as different entries; i.e. "project1" and "PROJECT1" are treated as two unique entries.)

The MacroType Electronic Typewriter makes MacroManager act as a pseudo electronic typewriter using margins, tab stops and justification. It allows lines as long as 70 characters to be

continued on page 273

Math Pack #2
continued from page 269

in a matrix. The matrix will have (number of unknowns + 1) columns and (number of equations) rows. If I call the array A, I can therefore refer to any number in symbolically by saying A[row][col]. (The upper left element of this array is usually referred to as A[1][1].)

The scale factor we need is computed:

$-(A[pivot][pivot]/A[row][pivot])$

In our case, the pivot is 1 and the row is 2. A[1][1] is 2 and A[2][1] is 1 so our scale factor is $-2/1$ or -2.

Now that we have the correct scale factor, we scale equation (row) 2.

$(-2) * 1 + 2 + 1 = 91$ is
$-2 - 4 - 2 = -182$

We now sum this equation with the pivot equation. (row 1)

$$\begin{array}{rl} & 2 - 1 + 3 = 68 \\ + & -2 - 4 - 2 = -182 \\ \hline & 0 - 5 + 1 = -114 \end{array}$$

(This is our new equation)

We now substitute the new equation for our old equation 2.

$2 - 1 + 3 = 68$
$0 - 5 + 1 = -114$
$2 - 1 + 1 = 46$

Now we are ready to "zero out" A[3][1]. We compute the new scale factor $-(A[1][1]/A[3][1])$ or $-2/2 = -1$. We scale equation 3:

$-1 * 2 - 1 + 1 = 46$ is
$-2 + 1 - 1 = -46$

Sum this with the pivot equation:

$$\begin{array}{rl} & 2 - 1 + 3 = 68 \\ + & -2 + 1 - 1 = -46 \\ \hline & 0 + 0 + 2 = 22 \end{array}$$

Now we substitute this new equation for the old equation 3.

$2 - 1 + 3 = 68$
$0 - 5 + 1 = -114$
$0 + 0 + 2 = 22$

In normal cases, we would have to "zero out" column 2 also, but in this case we already have the linear system in upper triangular form. Most times you would have to start with the pivot being 2 and go through the whole "zero a column" routine, then do it again with a pivot of 3 and so on until the system was in upper triangular form.

Once the system is in upper triangular form, we use backward substitution to solve the system. Solving for J is easy enough, we have $2J = 22$, so Jason's age is 11. Now that we have solved for J, we can subtract out J from the equations above it. In equation 2 we have $-5H + J = -114$. We multiply the coefficient of J times the actual value of J and subtract that from the constant. We then do the same thing for equation 1. We end up with:

$2 \quad -1 \quad 0 \quad = \quad 35$
$0 \quad -5 \quad 0 \quad = -125$
$0 \quad 0 \quad 2 \quad = \quad 22$

Now we can solve for another variable, this time H. We have $-5H = -125$ so $H = -125/-5 = 25$. We now subtract out H from the equations above it (only equation 1 in this case) and we get:

$2 \quad 0 \quad 0 \quad = \quad 60$
$0 \quad -5 \quad 0 \quad = -125$
$0 \quad 0 \quad 2 \quad = \quad 22$

Of course solving for B is simple, $2B = 30$, $B = 60/2 = 30$.

We now know that Jason is 11, Harry is 25 and Bill is 30. It always seems a little like magic to me that we could extract this from the information presented in the problem, but that is the power of linear systems.

This simple outline will give you the basic idea behind Gaussian Elimination and backward substitution. The source code for an implementation of this technique is presented in Listing 1. A few points about the listing need to be presented.

In C, all arrays start with element 0, not 1 as in many other languages. Keep an eye on your array numbers when you convert the program to another language. You will also see a lot of tests being done that will not make sense, this is because we have not yet discussed all of the different situations that can arise when dealing with linear systems. A better explanation of things that can go wrong, and how to get around them will be presented in the next article.

If there is interest in these programs, I will supply the source code and executable versions of the programs on 3½ disks for the Atari ST line of computers and on 5¼ disks for the IBM PC and compatibles. (Only the 360K drives for the IBM.) If you want the program but don't want to type it in, send $5 to:

Wynn Rostek
3456 Willis Drive
Titusville, FL 32796

Be sure and tell me you want the linear systems program and whether you need the IBM or Atari ST disk. ●

Wanderer

Reviewed by Jeff Brenner

Wanderer is a new action/adventure game, imported from France, for the Atari ST. The game's claim to uniqueness is its simulated three dimensional action. Not your standard 3D game graphics fare, mind you, where objects are merely drawn in a 3D perspective on the 2D screen, but full bodied, animated graphics that appear to move within the depths of your color monitor and sometimes even jump out at you from the screen. These aren't exaggerations, because Wanderer is used with 3D glasses, those same colored glasses used to watch the old 3D movies (non-colored polarized glasses have been the norm in the recent 3D movie resurgence).

Who Wrote This Plot?

Apparently, in their quest for originality and novelty, the authors have built the game around a plot so ridiculous that it is best ignored. What is the plot, you ask? In none other than 2986, cats are being kidnapped (yes, cats) and are slowly but surely disappearing from the face of the Earth. "Something must be done," the manual explains, "the cats must be found." Fortunately, the caretaker of the Shangri La Mansions, who lost her cat, found a playing card left behind by the kidnappers. The card leads you to "The Galaxy" where an interplanetary game of poker is being played. Your mission: Serve as a courier in "The Galazy" to bring cards from one planet to another and try to better each planet's hand. Your payment: Cats. But there's more nonsense splattered throughout this plot, including the "powerful ARK," which has been stealing the cats, black holes, and of course the SPHYNX, which you must ultimately battle after you've collected 8000 cats or five aces.

Playing The Game

Stripped of its plot, the game is more or less of a shoot-em-up, but probably one of the most exciting shoot-em-up games you've ever played. The three dimensional effects, once you can see them, are superb. The objects are simple

continued on page 278

MacroManager

continued from page 272

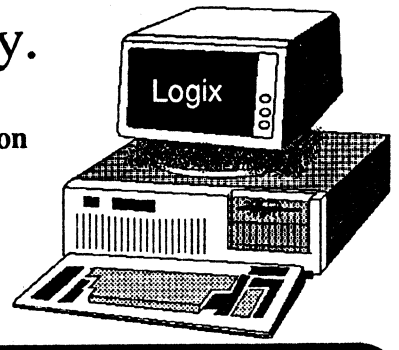
typed to the printer directly from the keyboard.

Anomalies and quirks. MacroManager appears to act erratically when used with certain desk accessories and/or programs; most noticeably with ARC.TTP and Thunder. If you make heavy use of the card file be prepared to start running out of memory if you have a 520ST. Also, I noticed that (like many other programs) it likes to alter the screen color on invocation, forcing you to pull the Control Panel down to reset the palette properly. Finally, it would have been nice if MacroManager would allow you to specify a directory to place all of the files it uses (a la Thunder) as opposed to defaulting to the root volume (for the card file, events and alarms) or the current working directory (for notes and plan files.)

Summary. MacroManager is a nice little desk accessory that deserves to be looked at and for someone who needs the kind of functionality that MacroManager has, it would be well worth the investment. ●

Put some **HIGH PERFORMANCE** on your desktop...

without sacrificing IBM* compatibility.



Finally, you can have full

- HARDWARE
- SOFTWARE
- COMPONENT
- DATACOM

compatibility certification

...and still get an...

UP TO 800% speed increase.

At Logix Microcomputer, we take a "system" approach to providing truly IBM* compatible computers. We offer four levels of compatibility: hardware, software, component, and datacom. This higher level of compatibility (or lack thereof) becomes apparent when you expand your machine to take on new tasks (beyond Flight Simulator* and Lotus 123*.)

Every computer ordered is configured the way you want, all options are installed (including hard drives) and then the system is put through a rigorous battery of tests, ending with a **96-hour (4 day) burn-in** session. All orders include a 9 month (270 day) limited warranty, **200 program library**, personal Tech Support Passcode, and Networking Newsletter.

Call the 24 hour, 7 days a week, convenience line: 1-203-877-6991

Introducing...

Logix™ **XTurbo2**

The highest compatibility of any "turbo" clone available today. 8MHz system clock effectively doubles the speed of the original IBM PC XT*.

You get: 512K RAM, silent 360K floppy drive, AT style keyboard, 8 expansion slots, 150W power supply, room for 3 more drives, 96 hour burn-in.

Add $99 for second floppy drive, and $29 for upgrade to 640K RAM.

$499

WOW! Logix™ **XTurbo3**

Watch out IBM! 10 MHz CPU with NO WAIT STATES, and super-fast access dynamic RAM, make this the fastest "turbo" system on the block.

The standard equipment, which includes 512K RAM, 3.5" 720K micro-floppy drive, and Enhanced Graphic Adapter, puts you solidly into the future of microcomputing.

Add EMS board, math chip, and hard drive for a multi-tasking Power-station! Enhanced model will drive virtually any hi-res monitor!

$595 Base Model **$895** Enhanced Model

EXPANSION OPTIONS

Monographic card.....$79.
720x348 graphics, crisp text, w. par. port
Color Graphic card.....$79.
640x200 graphics, w. par. port, RGB & comp
Enhanced Graphic card...$229.
640x350, 16 colors, 256K, smooth scroll
Hi-res monitor,tilt base...$125.
720x350, amber/green, beautiful text. Tops.
Enhanced Color Mon..$399/499
640x350, .31" dots for crisp text. Top quality
XTC Clock I/O card.....$75.
clock/calendar, serial, parallel & game ports
XTC 21M Hard disk kit..$399.
XTC 31M Hard disk kit..$449.
1/2 height, with controller & cables
ATC Serial/Parallel card...$75.
includes 9/25 adapter cable. 2nd ser. port opt.
ATC 3 meg Ram card.....$149.
No RAM- expand as you like. 16-bit card
256K chip set.....$33.
ATC 31M Hard drive.....$599.
ATC 42M Hard drive.....$699.
ATC 80M Hard drive....$1099.
26-39 ms, full height, auto-retracting heads
80287 w.10mHZ crystal..$395.
Blazingly fast Lotus & Autocad, w. reset sw.
3.5" 720K int. drive.....$149.
"Real" PC DOS package...$99.
IBM PC DOS 3.2, Adv. Basic, PC Tutor, man.

The compatible classic.

Logix™ **XTC**

Field proven in data communications: LANs, WANs, micro-to- mainframe, etc. Excellent network work-station. Buy this computer if you need the ultimate in compatibility confidence.

512K RAM, silent 360K floppy drive, "AT" style keyboard, 150W switching power supply, 8 expansion slots, room for 3 more drives, 96 hour burn-in. Add $29 for 640K, $99 for second drive.

$489

Most Powerful of all.

Logix™ **ATC**

Mini-computer capability. 10 MHz 80286 cpu, for maximum performance without compatibility compromises.

Excellent network server, or single-user power station. Handles massive hard drives, XENIX*, 3Com*, Novell*, IBM PC Network*, Autocad*, etc.

Standard equipment: 640K RAM, 1.2M floppy drive, 200W of power, famous keyboard, heavy-duty case, 16-bit hard-drive controller. See ATC drive options for your storage requirements.

$1299

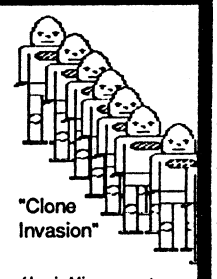
Computer Shopper Special !!

Order "Turbo Package." Includes: XTurbo2 computer system, complete with: 640K RAM memory, 2 silent 360K drives, clock/calendar, serial port, parallel port, 720x348 graphics display card, hi-res green or amber monitor with tilt & swivel base. Ready to run, just plug in and enjoy!

$795

VISA *Limited time offer. PC DOS 3.2 optional.* MasterCard

Logix™ *MICROCOMPUTER*
TEL. 203-877-6991
P.O. BOX 3574, DEPT CS17
131 NAUGATUCK AVE.
MILFORD, CT 06460
TELEX: 6503078990
MCI MAIL: 307-8990
Nat'l Sales Mgr: Tel. 203-783-1621



©1986 Logix Microcomputer *registered trademarks Shipping and Insurance billed at exact cost Logical and Logix logo are trademarks of Logix Microcomputer.

Rumors End
continued from page 123

who are offering software that should run on the C-128. Excaliber is communications program that is being sold as a C-128 CP/M program by Niagara Communications. They've been running a small ad in *Computer Shopper* for awhile, but in case you might have missed it, I thought I'd mention it here. It's $49.95. As usual the address is at the end of this article.

Roy Lipscomb of Logic Associates wrote to tell me that they are offering software in CP/M format. His company offers a free catalog and reviews of their software. He suggests that C-128 users order the Kaypro II or IV format.

Though I have not tried the programs, I will briefly describe them for the adventurous. They offer Columns, a program to manipulate text, similar to the page layout software that is offered with some of the desktop publishing packages. It offers cut-and-paste to move the text around. Price is $29.

Diskguise is a file handling routine that gives more flexibility to CP/M. Price is $32. Supermit adds flexibility to BATCH jobs. If you don't know how to use BATCH commands, this one won't mean much to you. Price is $39. Look for the ads in *Computer Shopper* for full descriptions of these.

The one that looked most useful to me, was the VersBase program. It asks for a number whenever a program is about to create a new version. If you answer yes, it will append the next highest version number to the file. For someone like me who goes through 5 or 6 copies when creating a file, this would be ideal. Price is $32

As usual *Computer Shopper* has forwarded my mail and I've had several interesting letters. This one has a problem I think I can help with.

Dear Ms. Peterson,

I would like to thank you for your C-128 CP/M column. I have been reading it since it started. Before I was a bit nervous about using CP/M and had little interest in CP/M so I didn't read very much of the 128 System Guide on CP/M. Your articles informed me that my 128 will run other formats such as Kaypro IV. Upon reading this I contacted a friend who is the head of programming at a realty company. He had made me an offer of free software when I bought my 128, but I turned it down because it was for Kaypro IV. The software was available because his company would write its own packages for its realtors on the Kaypro and did not want to give them the bonus package they received when they bought the Kaypro IV's. This package included WordStar, MailMerge, Word Plus, Microsoft BASIC-80, DataStar, ReportStar, CalcStar, and the main reason for this letter dBase II. I thought you might be interested in this after your articles in August and September and might want to pass it on to others. I called a couple of local Kaypro dealers and found out that dBase II is still available for Kaypro IV in CP/M for about $90.

I also have two questions for you, first when I try to make a working copy of these disks I cannot because PIP is not on these disks and when I use a command such as PIP E: = A:DBASE.COM it looks for the file on that disk. Any help would be appreciated. Second, how good is the documentation for the CP/M package that is offered in the Commodore 128 System Guide (found between 220 and 221) and would this possibly answer the first question?

Sincerely,
Robert E. Codd II

Thanks for writing Robert. I think I can help with your problem, but first I'd like to make a comment. I'm glad you turned down the offer of "free software." While the reason you gave in the letter was acceptable, I would have been a lot happier to hear you turned it down because it would also have been "illegal software." As you appear to have bought a legal copy of dBase II, I will spare you the lecture on software piracy and just answer the question.

Your PIP problem seems strange to me. Before trying to PIP the files, be sure you have a copy of PIP.COM on the destination disk. Put the destination disk in the drive and type your PIP command. You may need to type E:PIP E: = A:DBASE.COM. The disk drive should whirr a second or two and then the computer should prompt you to remove the E: disk and insert the A: disk. From there it should prompt you to switch disks a couple times. It should give you back the A > prompt when you are done.

The Digital Research Manual does have a lot of information and probably would answer your question. But there are less expensive ways to get an answer. If you are interested in really finding out how CP/M works, the Digital Research book may be the best way to go for you. (I understand from a source at Commodore that it has sold over 50,000 copies.)

Hope this helps you.

Regards from a previous writer Ted Chidester wrote several months ago looking for a recommendation of a good COBOL compiler. He writes again with further information on what he has found.

Cheryl,

The inquiry I sent you was part of a BATCH (inquiry about COBOL). Strangely, of the many possibilities, most

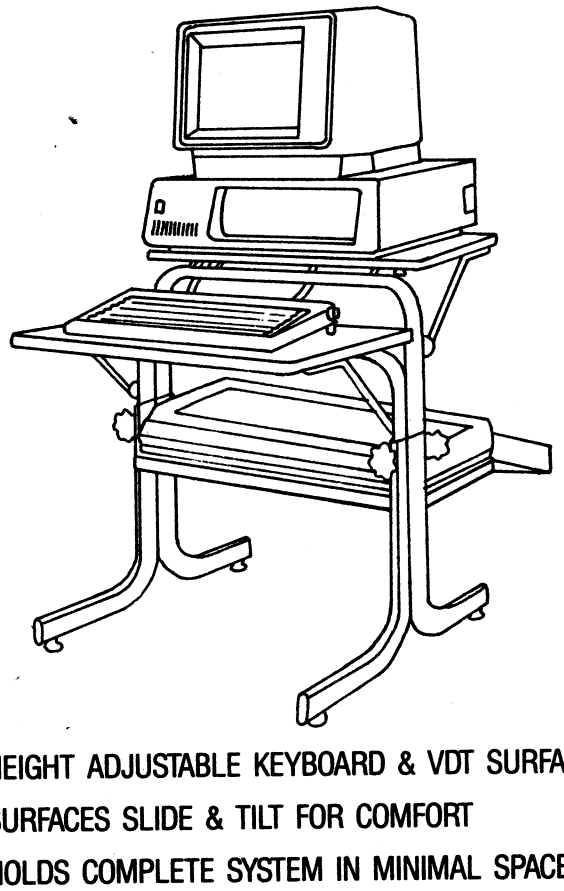
continued on page 280

Getting The Most
continued from page 124

add footnote capability, I am fairly confident that a home or college user will not outgrow them. The one weak link in these programs is the glacial slowness of Data Manager 128. I can help you speed it up and will make a few suggestions for future improvements to Timeworks next month when I tell you how to get the most from Data Manager 128.

I use several word processors on several machines so I don't try to remember all the various commands. I have often wondered why companies don't list the commands according to what needs to be done instead of what the command is to do it. Because of this, I make a list of commands as I study each new word processor. That makes it easy to look up what I want to do later. By the time I finish these commands, I have the program under control. ●

COMPUTER TABLE FOR HOME OR OFFICE



- HEIGHT ADJUSTABLE KEYBOARD & VDT SURFACE
- SURFACES SLIDE & TILT FOR COMFORT
- HOLDS COMPLETE SYSTEM IN MINIMAL SPACE
- NO TOOLS REQUIRED FOR ASSEMBLY

RETAIL $279.00 **OUR PRICE $139.00** + $6.00 Shipping

SEND CHECK OR MONEY ORDER TO: **COMPRO**

P.O. BOX 235, LONDONDERRY, NH 03053 — OR CALL

FOR C.O.D. SHIPMENT — **800-221-5399**

Wanderer
continued from page 273

undetailed shapes, but they move about the screen in all directions, toward and away from you, with outstanding three-dimensional realism. Shoot an object and you might find yourself ducking your head to avoid the oncoming debris.

The game uses shields, energy, and a sector map, much like the popular Star Raiders program for the 8-bit Atari. But the playing card concept is completely unique to Wanderer. By exchanging cards among the planets, you attempt to better the hands for payment in cats. Helping a planet gain a pair, for example, is worth 50 points; three of a kind is worth 300 cats; poker is worth 3000 cats; with a number of other combinations in-between. Again, it's silly stuff, but the game is very playable, aided enormously by the 3D graphics.

Seeing The 3D

Wanderer can only be played in 3D, since it looks like a tangled mesh of red and blue lines without the glasses. But some individuals will have trouble seeing the 3D, and others might not see it at all. The instruction manual gives a variety of tips for seeing the 3D best, such as turning up the contrast and darkening the room. Your enjoyment of the game will depend largely upon how well you can see the 3D.

The fourteen-page documentation booklet contains ample instructions, and includes a section that explains how the 3D effect is created. The language is at times awkward and confusing, since it was apparently translated, but on the whole the instructions are detailed enough to be understandable.

Conclusions

Without a doubt, Wanderer's 3D effects are what makes the game interesting and enjoyable. The game really shows off the high-speed computation capabilities of the ST, particularly when numerous objects are flying around the screen at once, all in 3D. The novelty of the 3D graphics will diminish each time you play the game, of course, but it's still nice to have at least one 3D game in your software library to show off.

Requires: Atari 520/1040ST, Disk Drive, Joystick.

Authors: Beatrice, Jean-Luc and Frederic Langlois.

Distributed by: Eidersoft, USA, P.O. Box 5607, Glendale, AZ 85312-5607.

Ratings For Wanderer: * Poor, ** Fair, *** Good, **** Excellent.

Program Design ***
Playability ***
Documentation ***
Average ***

IF YOU want to make money with your computer then you need our new book—How to Get Rich with your Microcomputer. Rush $14.00 postpaid to: **James Jones,** PO Box 32, Woodlawn, VA 24381.

GOOD LUCK FROM MR. Z.

FANTASTIC DAILY NUMBER FORECASTER!

Guaranteed Winners or Money Back! Picks up to 3 STRAIGHT WINNERS most every week, playing 1 to 3 a day! Apple, IBM, C64, Atari, 1 drive. Many reports of hitting for THOUAND. Hundreds sold. Guaranteed Winners See our large ad under Commodore Listing 113B. **Z-WAY Publications.**

Mnematics Videotex

"**MNEMATICS VIDEOTEX** epitomizes the legend of building a following through technical prowess and innovation.'—Russ Lockwood, A + Magazine. With our Apple, Macintosh, and Crab Apple SIGs and program databases, we're the online service for YOU. Local access in over 600 cities in USA, 120 countries. Call 1-800-322-3633 (**914-365-0184** in NY) voice to subscribe. 02-87

.. For an offer you can't refuse ...

GODFATHER'S used computer SYNDICATE

$$$$ New Human Edge Software at an incredible price!!! Retails for $250 — Wholesales for $150 = * = * = * = Our price only $35 delivered! For Sales Edge, Negotiation Edge, or Management Edge from the folks who brought you Mind-Prober!!!! Flashcalc by Visicorp who brought you Visicalc, incredible buy at $35!!! **GODFATHER'S USED COMPUTER SYNDICATE,** PO Box 3037, 851 W. Hwy. 436, Forest City, FL 32714. **305-774-1111.**

MACINTOSH SALE!!! Copy II $20; Davinci Buildings $20; MacLotto $20; MacLightning $40; MacPublisher $50; MSBASIC $50; 400K Drive $125. All used. Excellent Condition!! **A&S Software/cs,** 623 N. Lincoln, Lakeview, MI 48850. 03-87

★ ★ ★

MAC PEOPLE

★ ★ ★

CHEAP SOFTWARE

Mac Write/Paint (List $195) **$45**
Overview (List $295) **$45**
PFS File/Report (List $175) **$40**
Mac Chart (List $125) **$25**
Call CRAZY RICHARD! Shreve Systems 845 Lark Ave., S. Port, LA 71105.

318-865-6743
4—9 P.M. C.S.T.

03-88

MANUALS & PRODOS

Apple Originals: Unidisk 3.5" Owner's Manual with DOS, ProDos 3.5" Utilities disk $15. Similar, 5.25", no disk $5. //e 80 Column Card Manual with Extended Supplement, 131 pages, $12. II +, //e Memory Expansion Card Manual (256K—1 MEG), $5. "Apple II Utilities Guide", = ProDos and its 64K and 128K utilities, disk, 150 page manual, $35. Apple //e (enhanced) Owner's Manual with introductory disk, 192 pages, $18. Apple DOS 3.3 User Manual $16. Apple DOS 3.3 Programmer's Manual $12. $20 minimum order. Add UPS, $2 first, $1 each additional. Request our flyer. **Electrovalue,** Box 376-CM, Morris Plains, NJ 07950. **201-267-1117. 04-98**

MIDI SOFTWARE

For APPLE II, Macintosh. Sequencers, synthesizer editors, from Opcode, Sonus, Dr. T, Mark Of The Unicorn—all heavily discounted. MIDI interfaces from $65! Information: send SASE, specify your computer and/or synthesizer. **Livewire,** Dept. C-AP, Box 561, Oceanport, NJ 07757. Orders only call: **201-870-3115.** 03-87

Microsoft Premium Softcard for the Apple IIe. With manual and software $99. Call **Kelly Hancock 503-285-8876.**

UNUSED, UNOPENED software from store bankruptcy—BPI Accounts receivable V1.8 $45. Practicalc II $12. Human Edge's Management Edge $22. Crunch for Macintosh $16. Very limited quantities. Add $2 shipping. **R. Crocker,** 22101 123rd Ave. SE., Kent, WA 98031. 206-630-3388.

NOW ACCEPTING PUBLISHABLE PROGRAMS! Submit diskette, SASE to **ANTHONY's,** Box 1022, Crawfordsville, Indiana 47933. Order our Apple Tax Preparer $6.66.

PRINTSHOP LOVERS 15 new fonts including a fancy Old English. Printshop must be modified with Printshop Companion to read from data disk, $17. Delivered. **JL Jensen,** PO Box 766, Provo, Utah 84603. 04-89

★ ★ ★

PUBLIC DOMAIN APPLE SOFTWARE

A great way to own software at an affordable price. Well over 1000 programs on 50 diskettes - 10 disks in each of 5 categories - Business & Finance, Games, Utility, Education, & Miscellaneous $5 per disk + maximum of $2 for shipping and handling. Send $1 for catalog refundable with order $5 for master catalog on disk. **Peel Industries,** PO Box 2112, Merritt Island, FL 32952. 03-88

Public Domain Programs - from $2. Many good programs, games, utilities, business, music, art, tutorial. Both DOS and Pro.DOS and Appleworks files. Send for catalog. **PD Software,** PO Box 13256-A, Houston, TX 77219. 03-89

RIBBONS: Star 10X/SG10, 3 pack $5.50 + $.50 S&H. **Cherrytree Software,** 464 Elmwood Drive, Mobile, AL 36609. 205-343-2770.

★ ★ ★

SHO' NUFF' APPLE STUFF

IIE cases w/keyboard $175
Mac Motherboards $185up
IIC Motherboards $195
64K/80 Col Bd. $50
II + Motherboards $85up
Imagewriter I $275
5 Meg Profile $250
II + Cases $50
Don't see it???? Call **Shreve Systems,** 845 Lark Ave., Shreveport, LA 71105. **318-865-6743** 4—9pm. You gotta be happy!! 03-88

TOTAL MUSIC Macintosh sequencer/interface, half price, casio CZ101. **212-838-7251.**

Trade Apple Software. Send your list, I'll send mine. Also Apple IIe 80 column card, best offer. **David,** Box 1204, Torrington, WY 82240.

Trade Software for Apple. Exchange your list for mine. State needs and interests. **Alan Grund,** 928 Farnham #4, Columbus, WI 53925. 03-87

(109) ATARI COMPUTERS FOR SALE

ATARI ST OWNERS

Free Details on Program and Information Exchange. Send SASE. Sample newsletter $3. **ASTUN,** 6157 South 700 West, Murray, Utah 84123. 12-87

ATARI

Power Without the Price

Atari 20 Meg hard disk drive $575
1040 ST Color Computer system ..$899
520 ST color computer system ...$779
Panasonic 1080i printer and ST cable$225
● Near letter quality in all modes
Newell 256K XL memory upgrade 800XL/1200XL $65
● Includes all chips and MSDOS software
ICD 256K XL upgrade (less RAM and software) $39
Crystal 256K Axlon Type Upgrade for 800 .....$119
ICD P:R: connection modem/printer interface $75
Modem & Printer cable set for P:R:/850 .....$30
Omniview 80 column upgrade $39
● Includes software-Specify computer
Avatex 1200HC Hayes Style Modem $119
Add 3% S&H (Min. $3) 8% APO/FPO/Air. 4% Visa/MC. Send SASE for price list.

Dyscom Computers
1279 Tumbleweed NE
Uniontown, OH 44685
(800) 821-1297 x520 orders, (216) 821-1297 Ohio.

★ ★ ★

8-BIT ATARI owners your favorite public domain software on disks—games, music, etc. Over 100 disks available. Send $1 for catalog to: **Marion's,** 140 E. 7 St., Suite 4E, New York, NY 10009.

ATARI

Power Without the Price

Atari 1040ST color system $950
Atari 520ST color system $775
Atari Disk drives .....Call
256K 130XE style memory for 800XL/1200XL $65
256K Axlon style memory for 800 .$125
Panasonic 1080i NLQ Epson/IBM printer $209
Atari 850 Printer/modem interface .$115
Omniview XL/XE 80 column video ..$39
Atariwriter Plus Wordprocessor (disk)$39
● All upgrades include chips and MYDOS software. Add 3% for shipping, 7% Air/APO/FPO, 4% VISA/MC. Send SASE for price list and info.

Dyscom Computers
1279 Tumbleweed NE
Uniontown, OH 44685
216-821-1297 Ext. 520 Ohio
800-821-1297 Ext. 520 Orders Only!

New high-quality 6' ST parallel printer cables $7.50, 2-5 $6.50, 6+ $5.50. Also available TIMEKEEPER by Navarone $39.95. COMPUTEREYES ST Color Digitizer $194.95. IL residents add 6.25% tax. Add $3 shipping per order (not per item). **MARS,** 15W615 Diversey, Elmhurst, IL 60126. 03-88

WANTED: Atari ST computers. Will buy your system(s), or trade our equipment for your ST. Call today! **MEGATECH, 203-634-4593,** PO Box 4213, Yalesville Station, CT 06492. 03-87

(110) ATARI SOFTWARE, PERIPHERALS

100 + Dial-Your-Match BBS modem number! $5. **BBS-List,** Box 6055, Burbank, CA 91510 or 300 Baud: **818-840-8066. 04-91**

★ ★

1040 ST VIDEO CONVERTER

Allows use of regular "green screen" video monitor with 1040ST LOW-RES and MEDIUM-RES modes. Colors translate to different shades of grey. Simple plug-in unit has VIDEO and AUDIO phono jack outputs. US-$26.95 or CDN $36.95. (1st class mail) **ESQUIMALT DIGITAL,** 820 Dunsmuir Rd., Victoria, BC V9A 5B7, Canada **604-384-2626. 04-88**

520ST owners!! Upgrade your single sided drive to double sided! Completely compatible with all ST software and peripherals. $95. **AIC,** 2600 Mar Vista Drive, Aptos, CA 95003. **408-688-5000, 1-800-242-7776. 8-90**

800/XL/XE Software: Games, utilities, applications, graphics, music, education, Print Shop icons and much more at tremendous prices for your Atari 8-bit computer. Send SASE for large list. **J. Edwards,** 138 South Main Street, Travelers Rest, SC 29690.

★ ★ ★

AMAZING Atari Public Domain Software! Over 110 Low cost Theme Disks available, or fill disk with programs you want. Send for free 8-bit Catalog Disk. **Bellcom,** Box 1043-C, Peterborough, Ontario, Canada K9J 7A5 04-91

★ ★

ATARI ST

10 different ST Public Domain disks, and each disk is packed with quality programs. Educational, recreational, business and utilities applications included. Software provided on quality SONY diskettes and 5 FREE blank SONY diskettes included with every order. Order from the best—you will be pleased and you will reorder. Other offers and complete ST Listing included. ONLY $49.95. For VISA/MC orders ONLY, call **1-800-862-2345** (Wait for Dial Tone) then dial **"PAT".** Same day shipping. 2 DISK sampler for $10.95. For inquiries or mail order send to **PATNOR,** PO Box 370782, El Paso, TX 79937. 04-92

AAAT LAST! Professional Handicapping Systems: Thoroughbred/Harness $29.95/Enhanced $49.95; Greyhound $29.95/Enhanced $49.95; Pro Football $39.95; Plus $2 S&H. VISA/MC. Specify computer. **Softtech Group Inc.,** PO Box 582C, Keego Harbor, MI 48033. **313-851-4925.**

ATARI 800/XL/XE

Public Domain disks $3 per double sided disk. Utilities, Games, Music, Graphics, and more. Send SASE for list. **Brad Roltgen,** 719 E. Minarets, Fresno, CA 93710. **209-432-2159. 04-89**

ATARI ST OWNERS!

Here is the best selection of 520/1040ST Public Domain software. Over 40 disks in 8 categories available. Entertainment, Home/Business, Utility, Graphics, and more! Any disk only $5 each! Send business size SASE for catalog. **ST-Land,** PO Box 1646, Orange Park, FL 32067-1646. 03-88

ATARI ST

Public Domain disks $4 each; Utilities, accessories, games, graphics and more. Send SASE for list. **Brad Roltgen,** 719 E. Minarets, Fresno, CA 93710. **209-432-2159. 04-89**

ATTENTION ATARI 8-BIT'ers

Choose from the largest selection of XL/XE Public Domain software. Over 800 programs on nearly 100 disks available. Entertainment, Home, Business, Utility, Educational, more! And each disk full is only $5! Send business size SASE for catalog. Grab a "byte" today! **Gatorware,** Box 1215, Gainesville, FL 32602. 04-89

ATTENTION: $$WIN$$ with THOROUGHBRED/HARNESS, GREYHOUND Handicapping Software...$29.95, enhanced...$49.95. Professional **FOOTBALL** handicapper...$39.95. MC/VISA. Free information. **Software Exchange,** Box 5382CS, W. Bloomfield, MI 48033. **1-800-527-9467. 03-87**

ATR 8000 high speed disk drive adapter, printer port & serial port, 64K $200. **516-689-6873, Softsales,** PO Box 224, Kings Park, NY 11754. 04-88

★ ★ ★

Arcade Masters! Figure eight Auto Racing challenges you. Maximum skill required! $8/disk ppd. 130XE/XL/800/400 computers 48K no Basic. **Rascom,** 22128 Newkirk Ave., Carson, CA 90745. 04-92

Atari 825 printer graphics conversion kit. Send $39.95 to: **Dan Chun,** 4275 Queen Anne Dr., Union City, CA 94587. **415-471-9286. 02-90**

COMPUTEREYES $105. Color Computereyes for 1040ST, $210. Prices include shipping. To order send payment or call. Visa/Mastercard accepted. PA residents add 6% Sales Tax. **Seneca Software,** 1104 Seneca Drive, RD #3, Harmony, PA 16037. 412-452-8418.

CREATE NEWSLETTERS

Tired of hard to use typesetting programs? Want to mix pictures with text easily? You need THE NEWS STATION! Compatible with Print Shop (tm) icons, text files, word processors, etc. And what you see on the screen is printed out - TO SCALE! Requires 48k, Epson/Gemini/Panasonic or Prowriter printers. Also available Qwik Pix Atariwriter & Qwik Pix Paperclip. Each $26.95 + $3 shipping. **MARS,** 15W615 Diversey, Elmhurst, IL 60126. Need more info.? Send two stamps. 03-88

Find Out what your 800/XL/XE can still do. If you're a good Poker player, or want to be, then you'll want to run these 7-Stud and 7-Card No Peek programs. Only $4.55 on cassette. **GGS,** Box 542, Plainfield, NJ 07061-0542.

Free Atarist best of Public Domain Disk, when you buy any three disks. $4.95 each. SASE for list. **R. Fitzgerald,** Centerville Court E-62, North Syracuse, NY 13212. 04-89

★ ★ ★

KARATE

ST

A proven best-seller in England and Europe on other 68000 machines is now available here in the U.S. Incredible graphics, fast action, digitised sound. Write for **FREE CATALOG. Curry Computer,** PO Box 5607, Glendale, AZ 85312-5607. **602-978-2902.** A 24 hour TOLL-FREE order line is listed in catalog. 02-88

★ ★ ★

Game And Utility programs for the 8-bit Atari. Send $2 (refundable with order). **RWR Enterprises,** 3102 Bradford St., Woodbridge, VA 22193-1422. 02-88.

"Home Computer Users" Earn...$500—$5,000 monthly. Free Report! **Computime,** 8812-A Duncanson, Bakersfield, CA 93311. 03-88

GOOD LUCK FROM MR. Z.

FANTASTIC DAILY NUMBER FORECASTER!

Guaranteed Winners or Money Back! Picks up to 3 STRAIGHT WINNERS most every week, playing 1 to 3 a day! Apple, IBM, C64, Atari, 1 drive. Many reports of hitting for THOUAND. Hundreds sold. Guaranteed Winners See our large ad under Commodore Listing 113B. **Z-WAY Publications.**

Low cost public domain software. Send SASE for free info or $2 for info/demo disk to: **DarkSide,** PO Box 818, Plover, WI 54467-0818. 12-88

MoonSoft for the 8 bit Atarian. The Small Business System $25; LabelMaster V1.5 ONLY $7! All software on Disk. Send SASE for info. **Black Moon Systems,** POB 152, Wind Gap, PA 18091. 03-87

PRINTERS DELIGHT

Atari people, 2 disks, 100 different, high quality graphics on each disk, designed for Printshop. These are original. Stop paying inflated prices for low quality material. Send $8.50 for one disk, $16 both, to: **Printers Delight,** 505 Wildwood Drive, Mt. Zion, IL 62549. You will like these. 04-89

Public Domain 8 bit Software. Disk full both sides with quality games, utilities and business programs $4 each or 4 for $15 postpaid. SASE for list. **Douglas Puterbaugh,** RR #3, Box 449, Warsaw, IN 46580. 01-87

RIBBONS: Star 10X/SG10, 3 pack $5.50 + $.50 S&H. **Cherrytree Software,** 464 Elmwood Drive, Mobile, AL 36609. 205-343-2770.

Replacement Power Supply for Atari 400/800, 1200XL & 1050 disk drives $19.50. For Atari 800XL, 65/130XE $19.95; Atari/C64 6' monitor cable (5-pin din to 4 RCA plugs) $5.95 2-5 $4.95, 6+ $3.95. PR:CONNECTION $59.95. Disk notcher $5.95. Add $3 shipping per order. IL residents add 6.25% tax. **MARS,** 15W615 Diversey, Elmhurst, IL 60126. **312-530-0988. 03-89**

ST Software All kinds of Public Domain disks only $3 each. Buy, Sell, or Trade used software. Many ST programs available from our rental library. Send SASE for free list. **Compu-Soft,** Box 4782, Lancaster, CA 93539. 03-87

Service Manuals with Schematics for Atari 520ST, 800XL, 600XL, 1050, 400/800, or 5200 $17.50 each; RAM upgrade kits: Atari 600XL 64K $35; 800XL 256K $45; 130XE 320K $45..(All orders add $2 shipping). Call **805-927-4667** VISA/MC/COD or send to: **Electronic Dimensions,** PO Box 1846, San Luis Obispo, CA 93406.

USE THAT MIDI INTERFACE!

Make some music—we carry ATARI ST SOFTWARE by Hybrid Arts, Sonus, Steinberger, Dr. T, and more. ST computers, accessories too. MIDI music experts with Guaranteed LOWEST prices. For free newsletter send SASE to **Livewire,** Dept. C-ST, PO Box 561, Oceanport, NJ 07757 or call **201-870-3115.03-87**

VCR Catalog Program - ATARIST Software Specially designed file maintenance and printout program for organizing and cataloging VCR events - Epson printers - $19.95 or Free Catalog. **JHB Software,** PO Box 920952, Houston, TX 77292-0952. 04-88

★ ★ ★

WANDERER

ST

Already a No.1 best-seller in Europe on other 68000 machines. 3D graphics and 3D glasses create a "Hologram Effect" as your starship hurtles thru space half-way between your keyboard and your monitor. RGB color display required. Write for **FREE CATALOG.** We import the best from the U.K. and Europe. **Curry Computer,** PO Box 5607, Glendale, AZ 85312-5607. **602-978-2902.** Catalog lists 24 hr. TOLL-FREE number. 02-88